Deliverable D4.3

Final Architecture and Evaluation of BeGREEN O-RAN Intelligence Plane, and AI/ML Algorithms

for NFV User-Plane and Edge Service Control Energy Efficiency Optimisation

September 2025

Co-funded by the European Union

6GSNS

Contractual Date of Delivery:	**April 30, 2025**
Actual Date of Delivery:	**September 12, 2025**
Editor(s):	**Joss Armstrong (LMI)**
Author(s)/Contributor(s):	**Miguel Catalan-Cid, Esteban Municio, Miguel Fuentes, David Reiss (i2CAT)** **Juan Sánchez-González, Jordi Pérez-Romero, Oriol Sallent, Anna Umbert (UPC)** **German Castellanos, Simon Pryor (ACC)** **J. Xavier Salvat, Jose A. Ayala-Romero, Lanfranco Zanzi (NEC)** **Joss Armstrong (LMI)** **Jesús Gutiérrez (IHP)** **Alejandro Blanco (Telefonica)** **Mir Ghoraishi (GIGASYS)**
Work Package	**WP4**
Target Dissemination Level	**Public**

This work is supported by the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe research and innovation programme under Grant Agreement No 101097083, BeGREEN project. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or SNS-JU. Neither the European Union nor the granting authority can be held responsible for them.

Revision History

Revision	Date	Editor / Commentator	Description of Edits
0.1	4.12.2024	Joss Armstrong (LMI)	TOC first draft
0.2	13.5.2025	Joss Armstrong (LMI)	Chapter 2 ready
0.3	14.5.2025	Juan Sánchez, Anna Umbert (UPC)	Completed section 3.3.
0.31	20.5.2025	J. Xavier Salvat, Jose A. Ayala-Romero, Lanfranco Zanzi (NEC)	Section 3.5 ready
0.32	20.5.2025	Miguel Catalan-Cid (i2CAT)	Sections 2.1 and 2.3 revised
0.4	20.5.2025	Joss Armstrong (LMI)	Chapter 3 ready
0.5	19.6.2025	Alejandro Blanco Pizarro (TSA)	Revised section 2.1
0.6	15.7.2025	Jesús Gutiérrez (IHP), Mir Ghoraishi (GIGASYS)	Full Technical Review
1.00	12.09.2025	Simon Pryor (ACC)	Submission to the EC

Table of Contents

List of Figures

List of Tables

List of Acronyms

3GPP	3rd Generation Partnership Project
AI	Artificial Intelligence
AoA	Angle-of-Arrival
AP	Access Point
API	Application Programming Interface
B5G	Beyond 5G
BNR	Benchmark No Relays
CAPEX	Capital Expenditures
CDF	Cumulative Distribution Function
CH	Coverage Hole
COTS	Commercial Off-The-Shelf
CP	Control Plane
CPU	Central Processing Unit
CSI	Channel State Information
CSV	Comma-Separated Values
CU	Central Unit
DDPG	Deep Deterministic Policy Gradient
DL	Downlink
DME	Data Management and Exposure
DPDK	Data Plane Development Kit
DQN	Deep Q-Network
DT	Digital Twin
DU	Distributed Unit
DUT	Device Under Test
E2SM-KPM	E2 Service Model – Key Performance Measurement
E2SM-SSC	E2 Service Model for Slice-Specific Control
E2SM-SSM	E2 Service Model for Shared Spectrum Management
E2AP	E2 Application Protocol
ES	Energy Saving
FEC	Forward Error Correction
FRAO	Fixed Relays Always ON
FROO	Fixed Relays On/Off
FRR	Fixed Relays and RUEs
gNB	gNodeB
GPU	Graphics Processing Unit
ICS	Information Coordination Service
IMC	Integrated Memory Controller
IP	Internet Protocol
KPI	Key Performance Indicator
KPM	Key Performance Metric
L2	Layer 2
LLC	Last Level Cache
L1M-UL-SRS-RSRP	Layer 1 Measurement - Uplink Sounding Reference Signal - Reference Signal Received Power
LTE	Long Term Evolution
MAC	Medium Access Control layer
MAE	–
MCS	Modulation and Coding Scheme

ML	Machine Learning
mmpc	marginal memory power consumption
MNO	Mobile Network Operator
MPKI	Misses per 1000 instruction
MSE	Mean Squared Error
near-RT	near Real-Time
NFV	Network Function Virtualisation
NIC	Network Interface Card
NN	Neural Network
non-RT	non Real-Time
NSA	Non Standalone
NUC	Next Unit of Computing
O-RAN	Open RAN
PDCP	Packet Data Convergence Protocol
PHY	Physical layer
PMD	Poll Mode Driver
RAN	Radio Access Network
RUE	Relay UE
PoC	Proof-of-Concept
PS	Power Saving
QoS	Quality of Service
RAN	Radio Access Network
RAPL	Running Average Power Limit
RF	Radio Frequency
RFE	Recursive Feature Elimination
RIC	RAN Intelligent Controller
ROP	Result Output Period
RSRP	Reference Signal Received Power
RU	Radio Unit
RUE	Relay User Equipment
SDG	Stochastic Gradient Descent
SHAP	SHapley Additive exPlanation
SINR	Signal to Interference and Noise Ratio
SLA	Service Level Agreement
SNR	Signal-to-Noise-Ratio
SVM	Support Vector Machine
TDP	Thermal Design Power
TGW	Telemetry Gateway
UE	User Equipment
UL	Uplink
UPF	User Plane Function
vBS	Virtual Base Station
VPP	Vector Packet Processor
vRAN	Virtualized RAN
WP	Work Package

Executive Summary

This document, BeGREEN D4.3, consolidates BeGREEN's final results on AI/ML-assisted energy efficiency for 5G/6G-ready RANs. It refreshes the Intelligence Plane introduced in BeGREEN D4.2, quantifies the AI/ML plane's own energy footprint, validates closed-loop control with O-RAN components, and reports end-to-end gains across vRAN, live 5G NSA, relay-enhanced RANs, and edge scenarios hosting UPF and AI services.

Chapter 2 revisits the Intelligence Plane, keeping the core AI Engine with near/non-RT RIC integration while adding hooks for CF-mMIMO and ISAC. Interfaces from BeGREEN D4.2 are preserved; new policies and telemetry enable sensing-assisted control and user-centric coordination. We analyse the energy cost of the AI/ML plane itself—training, serving, storage, orchestration—against the network-side savings it drives, showing that BeGREEN's lightweight models achieve favourable trade-offs between size, accuracy, and inference rate. Final validations benchmark serving workloads, automate dataset creation and model training with energy metering, and verify AI Engine–RIC workflows on a representative cell on/off loop, including conflict management procedures. Additional scenario validations will be reported in BeGREEN D5.3 as part of PoC#1.

Chapter 3 presents the final evaluation of five AI/ML-assisted methods to enhance energy efficiency. In vRANs, we study cache isolation and LLC utilisation and introduce MemorAI, a lightweight controller that steers cache/CPU to active threads, lowering platform power without harming throughput or latency. For carrier on/off, a compact classifier uses traffic/context features to trigger sleep/activation with guardrails, delivering energy savings without SLA regressions in offline and lab tests. In relay-enhanced RANs, we define RUE identification, compare activation policies, and quantify coverage, sum-rate, and uniformity gains versus no-relay baselines, including detailed power consumption. For UPF, we characterise compute–power curves, validate experimentally, and build a traffic-aware ML policy for scaling and placement that reduces energy on realistic traces at constant QoS. Finally, we jointly orchestrate vRAN and edge-AI services with a multi-objective coordinator that maintains QoS while cutting total energy in static, heterogeneous, and dynamic scenarios.

Chapter 4 concludes that BeGREEN delivers an O-RAN-aligned Intelligence Plane extended for CF-mMIMO/ISAC, practical guidance to ensure the AI plane yields net energy savings, and validated AI-driven procedures that reduce energy without QoS penalties. The accompanying assets—configurations, policies, measurement scripts—are ready for replication in PoCs (D5.3) and for transfer to standardisation and industry pilots.

1 Introduction

The transition to Beyond 5G (B5G) and 6G networks has underscored the urgent need for sustainable solutions to mitigate the rising energy consumption of next-generation communication systems. As network operators strive to meet the growing demands for ultra-reliable low-latency communications, massive connectivity, and enhanced data rates ensuring energy efficiency has become a critical priority. The BeGREEN project adopts a holistic approach to evolving radio access networks (RANs) that not only accommodate increasing traffic and service levels but also consider EE to meet global greenhouse gas emissions reduction targets set by the EU and other countries around the world.

BeGREEN D4.3 comprises the final evaluation of the efforts in BeGREEN Work Package 4 (WP4), focused on developing and validating AI/ML-driven solutions aimed at optimizing energy efficiency across the RAN, Core, and Edge domains. This deliverable builds on the two previous deliverables produced by WP4: BeGREEN D4.1 [1] reviewed the State-of-the-Art (SotA) and challenges, and presented the initial definition of the BeGREEN O-RAN Intelligence Plane and the proposed AI/ML-driven methods to enhance energy efficiency. BeGREEN D4.2 [2] described the progress on the development of the proposed solutions and presented their initial validation.

The BeGREEN Intelligence Plane, central to this framework, continues to evolve as a cross-domain management entity that integrates advanced AI/ML technologies to monitor, analyse, and optimize network operations. By leveraging the Intelligence Plane's AI Engine, Service Management and Orchestration (SMO), and RAN Intelligent Controllers (RICs), BeGREEN addresses the multifaceted challenges of EE. These challenges include the management of dynamic network environments, the efficient utilization of virtualized RAN (vRAN) resources, and the incorporation of emerging technologies such as Reconfigurable Intelligent Surfaces (RISs), Integrated Sensing and Communication (ISAC), and Relays.

BeGREEN D4.3 expands on the solutions and methodologies introduced in prior deliverables and focuses on their refinement, integration, and comprehensive evaluation. It details advancements in AI/ML model development and optimization, the deployment of energy-saving strategies, and the validation of these solutions in realistic and controlled scenarios. This deliverable also explores the joint orchestration of RAN and Edge AI services, the management of energy-efficient user-plane functions, and the enhancement of data-driven decision-making processes.

The document is structured as follows:

- **Chapter 2** elaborates on the final architecture of the BeGREEN Intelligence Plane, including enhancements to its components, interfaces, and functionalities. This chapter emphasizes the integration of advanced energy metrics such as Energy Score and Energy Rating and details the incorporation of cutting-edge technologies like RIS and ISAC. Finally, the functionality of core procedures and components is validated.
- **Chapter 3** provides a thorough evaluation of the proposed AI/ML-driven solutions, showcasing their performance in addressing key energy-saving challenges. Key focus areas include vRAN resource allocation, QoS-aware cell on/off switching, intelligent relay deployment, traffic-aware compute resource management, and joint orchestration of RAN and Edge services.
- **Chapter 4** summarises the findings and key insights from the evaluations, highlighting the impact of BeGREEN's solutions on improving energy efficiency across network domains.

Through the advancements described in WP4 and evaluated here, BeGREEN reaffirms its commitment to driving innovation in energy-efficient network design and operation. By addressing critical challenges and demonstrating the feasibility of its solutions, BeGREEN paves the way for sustainable, scalable, and intelligent B5G and 6G networks that balance performance and environmental responsibility.

2 BeGREEN Intelligence Plane

The main architecture of the Intelligence Plane, including its main components and interfaces, was presented in BeGREEN D4.2 [2]. While most of the architecture remains largely unchanged, focusing on the integration of the AI Engine and the RICs, additional analysis have been conducted to incorporate novel technologies investigated in the BeGREEN WP2 and WP3, such as cell-free massive Multiple-Input Multiple-Output (CF-mMIMO) and ISAC. The updates to the architecture are presented in Section 2.1, along with a summary of the complete Intelligence Plane architecture described in BeGREEN D4.2 [2].

Section 2.2 focuses on analysing the energy consumption of the AI/ML plane itself. With the rise of AI/ML, significant effort has been dedicated to developing AI/ML-driven optimizations, supported by models with diverse characteristics. This is also the case for BeGREEN, whose ML models aim to enhance the energy efficiency of beyond 5G and 6G networks. However, in this context, it is also relevant to assess the energy efficiency of these models, exploring the trade-offs between energy consumption and model performance and utility. Notably, the BeGREEN models presented in Chapter 3 do not require heavy training or high resource consumption due to their characteristics. Nevertheless, Section 2.2 aims to provide insights and clarifications into this topic.

Energy consumption is also considered in Section 2.3, which presents the final validations of the Intelligence Plane. First, complementing the benchmark presented in BeGREEN D4.2 [2], the energy consumption of the Intelligence Plane is evaluated based on the workload generated by serving model outputs. Additionally, validation of model training is presented, focusing on energy consumption and the automation of dataset creation. Then, the interaction between the AI Engine and the RICs is validated by presenting the main workflows and procedures involved in the BeGREEN AI/ML-driven control loops, using as example the cell switching on/off use case. Finally, validation of the conflict management procedures is discussed. Further validations will be included in BeGREEN D5.3 as part of the use cases defined within proof-of-concept 1 (PoC1).

2.1 Intelligence Plane Architecture

Figure 2-1 illustrates the final architecture of the BeGREEN Intelligence Plane. As highlighted in previous deliverables, the proposed architecture addresses two main objectives. First, to provide a serverless execution environment hosting the AI/ML models through the AI Engine, which provides inference and training services for rApps/xApps, effectively integrating AI/ML into the O-RAN architecture. Second, to incorporate the necessary interfaces and components to manage the RAN and Edge technologies, supporting BeGREEN's energy efficiency optimizations. Thus, this architecture represents BeGREEN's final design, integrating its key innovations and contributions.

Next subsections detail the concepts and implementation of the main building blocks of the architecture.

2.1.1 Intelligence Plane baseline

The Intelligence Plane baseline is formed by the main components enabling programmability and intelligence in the O-RAN architecture, i.e., the Non-RT RIC within the SMO and the Near-RT RIC, plus the BeGREEN AI Engine. As the complete architecture was already described in BeGREEN D4.2 [2], in this BeGREEN we provide a comprehensive summary of the key components, interfaces, and workflows to ensure a clear understanding of the validations presented in Section 2.3.

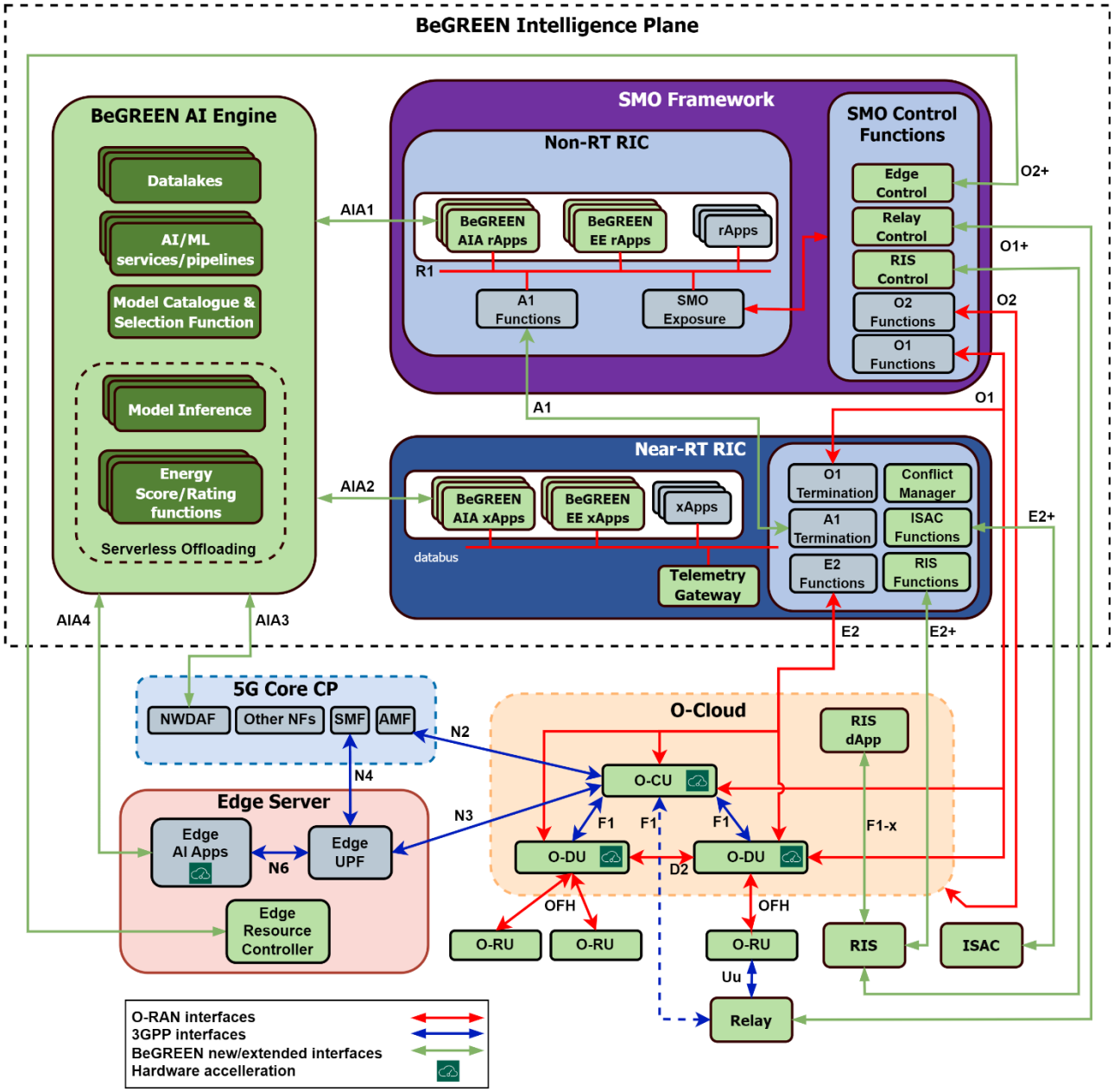


Figure 2-1: Final BeGREEN Intelligence Plane architecture

2.1.1.1 AI Engine

The AI Engine enables the implementation and exposure of AI/ML services in a loosely coupled manner. This means that models are trained and hosted in the AI Engine, with their inference exposed to rApps and xApps rather than being embedded directly within them. As noted in BeGREEN D4.2 [2], this approach promotes model reuse and allows ML and RAN control developers to focus on their respective areas of expertise, leveraging AI/ML and RAN management services as independent yet interoperable components. Additionally, the AI Engine framework implements serverless inference, enabling model offloading through serverless computing and hardware acceleration. This permits to leverage AI Engine benefits both in the non-RT and near-RT domains, as well also in other domains such as the 5G Core or the Edge services, as illustrated in Figure 2-1 through AIA-x interfaces.

Implementation-wise, the AI Engine has been developed using MLRun[1] and Nuclio[2] frameworks, which provide MLOps and serverless capabilities in Kubernetes environments. As detailed in Section 2.3, the main validation and evaluation efforts has been focused on AIA-1 interface for inference, connecting the AI Engine and the non-RT RIC. Nevertheless, similar approach can be applied to the near-RT RIC and the other domains.

The AI Engine also provides specific BeGREEN metrics, the Energy Score and the Energy Rating [2], which are used to determine the absolute and relative performance of the network entities in terms of EE, allowing to support BeGREEN optimisations and track their impact. The Energy Score and Energy Rating functions in BeGREEN calculate the EE in the network at the level of any component that measures both the volume of data that it transmits and its energy consumption. In addition, an Energy Measurement function enables the BeGREEN energy-saving ML functions to calculate their own energy consumption. This allows that the measured energy savings account for the additional energy overhead required to run these components, providing a net savings estimate. Additional details about these functions are provided in Section 2.2.1.

Additionally, the AI Engine includes a Model Catalogue and Selection Function. As described in Section 2.2.4, the objective of this function is to provide energy efficient model selection according to the required accuracy. AIA rApps and xApps can make use of this function to serve the appropriate model according to the request of the consumer rApps and xApps.

2.1.1.2 SMO and Non-RT RIC

The main role of the SMO in BeGREEN is to host the functions required to control and monitor RAN and Edge resources via O1/O1+ and O2/O2+ interfaces, which are exposed to control rApps. Therefore, in addition to the standard O1 and O2 functions defined by O-RAN for managing O-nodes and O-Cloud [3], it incorporates components and interfaces related for RIS, Relays and Edge resources. Details on these additional functions are provided in the following subsections, which focus on specific technologies.

Regarding the non-RT RIC, as was detailed in D4.2 [2], two main functionalities are implemented and validated. First, the life cycle management (LCM) of rApps, including: (i) control rApps devoted to cross-domain optimisations targeting EE, (ii) AI Engine Assist (AIA) rApps exposing the AI Engine services, and (iii) producer rApps exposing data such as Key Performance Metrics (KPMs). Second, the non-RT RIC implements the R1 interface to enable rApps to exchange data among them and to access the SMO (RAN and Edge control and monitoring) and the near-RT RIC (xApp management through A1 policies) services.

The key component to implement the R1 interface, particularly the Data Management and Exposure (DME) functions, is the Information Coordination Service (ICS) component, provided by the O-RAN Software Community (OSC) [4]. This component manages the available data types and the communication between data producers and consumers. In BeGREEN we also leverage this service to expose the inference outputs of the ML models, which are defined as data types and produced by the AIA rApps. In BeGREEN D4.2 [2] it was defined the model of the information types, the AIA rApps and the control rApps, which have been validated in the different demonstrations of the project[3] [4].

The final implementation of the non-RT RIC has also focused on the realization of the A1 interface to manage A1 policies related to Energy Savings. As will be detailed in Section 2.3.4, we have adopted the models presented in BeGREEN D4.2 [2], both in the Non-RT and Near-RT RIC sides, allowing to exchange A1 policies related to the operational status of the cells between the control rApps and the Energy Saving xApps.

[1] https://www.mlrun.org/
[2] https://nuclio.io/
[3] https://www.youtube.com/watch?v=_N0JY0Sepgc
[4] https://youtu.be/Iauc_-ffb8E?si=LrIlb0uCSdZKZ_mu

2.1.1.3 Near-RT RIC

The Near-Real-Time RIC in the BeGREEN architecture acts as the central action and decision-making entity for RAN control, hosting xApps that operate with under second-level responsiveness. It receives A1 policy guidance from the Non-RT RIC via the A1 interface and relies on timely, telemetry data from the RAN. To ensure a consistent and standardised data flow, the Telemetry Gateway (TGW) is a central component in the BeGREEN architecture that ensures seamless integration and interoperability across heterogeneous RAN infrastructures. Described in BeGREEN D4.2 [2], it collects, translates, and standardises telemetry data from multiple interfaces—including O-RAN compliant (E2, O1) and non-compliant sources—into a unified format for use by the Near-RT RIC and its xApps. By decoupling data processing from specific vendors or protocols, the TGW enables consistent metric generation, supports near real-time control actions, and allows advanced applications like energy saving and mobility management to operate effectively across diverse network environments.

Among the xApps deployed, the Energy Saving xApp and the Handover Manager xApp are key components of the BeGREEN system. The Energy Saving xApp dynamically switches cells on or off and adjusts power levels to reduce energy consumption based on telemetry inputs, traffic predictions and issued A1 policies, achieving substantial savings while maintaining QoS. In parallel, the Handover Manager xApp leverages advanced algorithms like Mobility and Load Aware proActive haNDover Algorithm (MOLA-ADNA) to optimise UE mobility across the network, outperforming traditional reactive methods through proactive, multi-metric analysis that includes RSRP, RSRQ, throughput, and cell load. These xApps are tightly integrated with the dRAX databus and the TGW, allowing seamless access to metrics and enabling coordinated control actions.

To ensure consistent system behaviour, in BeGREEN we have introduced a collaborative conflict mitigation framework within the Near-RT RIC. Conflicts between xApps or A1 policies—whether objective, resource-based, or scope-related—are addressed through detection, resolution, and avoidance mechanisms. A Conflict Manager entity tracks xApp subscriptions, manages policy distribution, and monitors the state of RAN elements through a centralised database. Each xApp includes a conflict avoidance handler that reacts to alerts and policies in real-time. This architecture allows the system to resolve control conflicts proactively, ensuring harmonious operation across the AI-driven control loops and enabling scalable, policy-aware orchestration for energy-efficient and intelligent RAN management.

2.1.2 Edge domain

As illustrated in Figure 2-1 and described in previous BeGREEN D4.1 [1] and D4.2 [2], BeGREEN incorporates into the Intelligence Plane the management of Edge domain resources. According to the O-RAN architecture, edge resources can indeed be considered part of the O-Cloud, containing virtualized RAN components such as the CU or the DU [5]. In this context, the work detailed in Section 3.1 presents an AI/ML-assisted method to address the problem of compute resource allocation in virtualized RAN under shared computing infrastructure. Additionally, in BeGREEN, we consider the dynamic management of edge resources dedicated to other functions closely integrated with the RAN, such as the User Plane Function (UPF) of the 5G Core or AI services. Details on the procedures and algorithms for EE in these domains are provided in Sections 3.1 and 3.4, while in this section we briefly summarize the main components, interfaces and functionalities that would be required in the O-RAN architecture to integrate the proposed solutions, as illustrated in Figure 2-1.

First, we consider an Edge resource controller, placed in the Edge, which similarly to the O-RAN Infrastructure Management Services (IMS) [6] should expose methods to allow the dynamic monitoring and management of edge resources. Specifically, the following services should be available:

- Inventory and policies: The Edge Resource Controller shall expose methods to retrieve server characteristics, including the pool of available resources (e.g., number and type of CPUs and GPUs) and available energy-saving policies (e.g., performance or energy-saving mode, supported P-states

and C-states, etc.).

- Monitoring: The Edge Resource Controller shall provide status updates, metrics, and alarms related to resource utilization and availability (e.g., CPU/GPU load, power consumption, energy usage). Consumers, such as the Edge Control Function located at the SMO, can subscribe to these metrics via the proposed O2+ interface. Combined with performance indicators of hosted functions, such as data volume processed by the CPU, energy-related metrics may be used to compute the Energy Score and Rating of the servers.
- Dynamic management: The O2+ interface shall expose functions and policies for dynamic resource management, including CPU/GPU allocation and configuration. Non-RT control shall be the minimum time granularity, though near-RT control could be beneficial in some cases (e.g., managing C-states).

In order to integrate these functionalities into the SMO, BeGREEN proposes (i) an Edge Control Function, which should implement similar functionalities as the Federated O-Cloud Orchestration and Management (FOCOM) manages, and (ii) a O2+ interface, working similar as O2-IMS [7]. Indeed, both the proposed function and interface could be incorporated into a broader specification of FOCOM and the O2 interface to support the general management of edge resources beyond the RAN, as highlighted in Sections 3.1 and 3.4.

2.1.3 Relays

The development of different relay control functionalities leads to the improvement of decisions of relay deployment, relay activation/deactivation, etc., which can improve the system performance and also provide a significant reduction in the energy consumption. BeGREEN has considered two types of relays: (i) the use of fixed relays to support 5G in the context of the recently Integrated Access and Backhaul (IAB) technology introduced by 3GPP [8], and (ii) the concept of UE-to-network relaying, in which a Relay-UE (RUE) has the capability to relay the traffic of another UE to/from the network in a two-hop communication manner [9]. Then, the relay represented in Figure 2-1 can be either an IAB-node or a Relay-UE. In case of a RUE, it is connected with the RU through the Uu interface. In case of the relay being an IAB-node, this interface Uu is also used to connect IAB-node Mobile Termination part (referred as IAB-MT) with the RU and, in addition, the interface F1 represented in dotted line in Figure 2-1 is used to connect the DU part of the IAB-node (referred as IAB-DU) with the CU part of the IAB-donor. The physical realisation of these interfaces is done on top of the Uu interface between the IAB-MT and the RU.

As introduced in BeGREEN D4.1 [1] and D4.2 [2], the considered relay control functionalities cover different aspects. On the one hand, a functionality for the identification of periods of time and geographical regions with high traffic demands and poor propagation conditions has been proposed in BeGREEN. On the other hand, another relevant relay control functionality is the identification of UEs that can be good candidate to serve as RUEs (i.e. UEs with relaying capabilities that may serve neighbour users with poor propagation conditions with respect to the gNB). Additionally, a functionality for the identification of the necessity to deploy fixed relays and the determination of the most adequate geographical location for their placement is also considered. Finally, BeGREEN considers the dynamic activation/deactivation of these fixed relays/RUEs. According to this, relays/RUEs are activated when needed to serve other UEs located in the coverage hole regions and are deactivated when they are not necessary, with the aim to minimize the energy consumption.

For the implementation of the previously mentioned relay control functionalities, several components and interfaces are necessary in the BeGREEN architecture, as described in detail in D4.2 [2]. On the one hand, a relay control placed at the SMO is in charge of the interaction with the relays for the collection of network measurements and the relay reconfigurations. This is done through an extended O1 interface, denoted as O1+ interface in the BeGREEN architecture. Measurements collected in the gNBs are sent to the SMO through the O1 interface. The different relay control functionalities are sustained by different rApps located in the Non-RT RIC. In particular, a Data Collection rApp oversees the management of the collection of

measurements in the RAN. Moreover, a Relay Management Function is in charge of the coordination and management of the different relay control functionalities. Each of these functionalities are sustained by different AI/ML models and databases hosted in the AI Engine. Finally, a set of AIA rApps are also necessary in the Non-RT RIC to cover different aspects such as data pre-processing, performance monitoring of the AI/ML models, model inference exposure and determination of the necessity of model updates or model retraining. More details of the architecture, functionalities, workflows and algorithms are provided in BeGREEN D4.2 [2].

2.1.4 RIS

Another key stone of BeGREEN EE strategies is the use of RISs. The ability to dynamically adapt RIS behaviour, encompassing decisions around their strategic placement and real-time configuration, unlocks substantial gains in system efficiency. As introduced in BeGREEN D4.2 [2] and D3.3 [10], RIS management is done by both the near-RT RIC and the non-RT RIC.

The control of the RIS is mainly required by coverage extension related use cases, where the RIS can reflect incoming RF signals from UEs and gNBs enabling communication in those scenarios where the signal is too weak, or increasing signal quality to improve the RAN efficiency. Such use cases usually require configuring the RIS with a pre-calculated codebook to reflect the signal towards the desired angles (azimuth θ and elevation φ). However, deciding which are the optimal angles is not trivial, especially in scenarios highly affected by multipath. Therefore, approaches such as fast beamsweeping or exhaustive search outside the codebook are normally required. For this reason, the RIS control channel should also support per-element phase configuration. In addition to this, RIS can also support advanced ISAC solutions to locate the user. Besides the benefit of obtaining positioning information, knowing the location of the user can improve EE by improving the communication channel, including optimized beamforming and coverage extension. For those localization use cases, RIS control may require supporting the configuration of the phase offset from a given codebook, e.g., to adjust the phase of the signal at the receiver and allow for Angle of Arrival (AoA) positioning. Integrating such RIS control mechanisms within the BeGREEN architecture requires RIS-specific components and interfaces, as presented in Figure 2-1 and previously discussed in BeGREEN D4.2 [2] and D3.3 [10]. At the SMO, the rApps handle the non-RT use of RIS (such as codebook training, non-RT coverage extension, position requesting, etc.), and the O1+ includes end points to directly interact with the RIS (see D3.3 for more details). On the other hand, the near-RT RIC will host the xApps involving the near-RT use of RIS (fine-grained configuration, mobility support, near-RT coverage extension, etc.), and will implement the RIS related E2SM, such as the E2SM-SSC and E2SM-SSM (see BeGREEN D4.2 [2] and BeGREEN D3.3 [10] for more details).

Finally, the BeGREEN architecture will also include the in-site control of the RIS through the F1-x interface and possibly dApps [11] located in the gNB. dApps may perform RIS+ISAC related actions (e.g., I/Q sample processing) or just become a proxy for the RIS xApps, offloading control overhead from the E2+ to the F1-x interface. Also, as discussed in BeGREEN D3.3, RIS related xApps may interact with the dApps through the E2SM-SSM and E2SM-SSC using the E2+ interface. We validate this RIS integration model into the BeGREEN architecture in following Section 2.3.6.

2.1.5 ISAC

The primary issue that arises when considering non-3GPP access networks as sensing information sources is how these can report their metrics through the 5G system in a standardised and efficient manner. BeGREEN intends to do this in a lean manner without involving the full 5G system but considering only the RAN part and the RICs. Given the work carried out in BeGREEN with a non-3GPP non-Wi-Fi node, such as the Sub-6 sensing system presented in BeGREEN D3.1, D3.2 and D3.3, it is proposed to extend the architecture to allow the integration of these “external” sensing nodes.

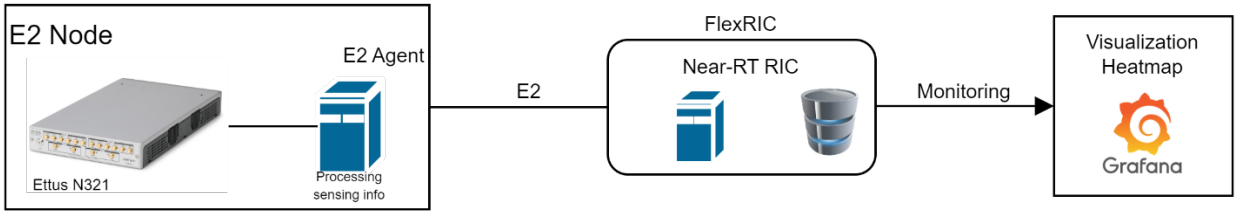


Figure 2-2: Proof of concept for sensing data exposure from non-3GPP wireless nodes to the RIC

The output of the sensing information from these devices is exposed to the RAN segment in a secure way through suitable extensions of the E2 interface of the Near-RT RIC. The E2 interface, which facilitates communication between the near-RT RIC and RAN nodes also employs SCTP over IP, with a suitable protocol for the communication over the E2 interface (E2AP) handling the control signalling at the application layer. An E2 Service Model (E2SM) for sensing is required to establish the communication between the sensing node and the RIC, and this communication is carried out through publication and subscription mechanisms to facilitate the exchange of information between xApps and RAN functions.

The messages defined by the E2AP protocol, also known as procedures, encapsulate different E2SMs. These messages enable the implementation of functionalities associated with both the control and monitoring of RAN metrics. From the final BeGREEN architecture shown in Figure 2-1, the ISAC node represented therein corresponds to the E2 node depicted in Figure 2-2, where the E2 Agent receives sensing data the sensing data processing unit. This processing is performed locally and generates sensing information, in the form of heatmaps that can be represented in a visualization tool, e.g. Grafana. In particular, a new sensing SM has been developed, which is integrated as a library in each of the agents involved: the near-RT RIC, the xApps, and the E2 agent.

2.1.6 Cell-free networks

A cell-free (CF) architecture is conceived as a user-centric architecture that aims to provide the best link quality at any time. To this end, distributed Access Points (APs) are pervasively deployed, and the best ones are selected to serve each of the user. Hence, a cell-free system requires coordination among the different APs, a joint processing, and dynamically adapting the selected of APs over time according to the position of the UEs and their needs. In the scope of O-RAN, O-RUs are the cell-free APs and the O-DUs compute the joint processing. A cell-free system makes use of the distributed APs to erase cell boundaries. However, in an O-RAN deployment, there are logical boundaries as one UE can be served by different O-RUs that are not connected to the same O-DU. We refer to this user as the edge user.

When serving an edge user, one O-DU will be the main O-DU and will receive user-related data from O-RUs that are connected to the other O-DUs, and the latter will share this data through the Inter-O-DU interface. Therefore, we need the O-DUs to cooperate and collaborate in executing the joint processing. This coordination is done by enabling an interface that connects different O-DUs as depicted in Figure 2-1. Although still not fully specified, this interface, denoted as D2, is currently under the O-RAN Alliance's standardization process [12].

Regarding the Near-RT RIC, it can determine whether the UE is a regular or an edge user and assign the best RUs to this UE. This process, which needs to be updated over time as the user moves, might be done by an xApp.

2.2 Energy efficiency in the Intelligence Plane

Since the goal of the BeGREEN project is to reduce energy consumption required to operate and manage radio networks, it is important to also minimize the energy consumed by the functions that BeGREEN

employs to achieve this objective. The energy consumed by energy saving functions must be measurable to provide comparable baseline measurements to calculate the savings achieved by BeGREEN energy saving functions. In the BeGREEN project, EE is calculated as the ratio of bits transmitted per joule of energy. The AI Engine hosts specific functions to facilitate the measurement and calculation of both absolute and relative EE metrics as described in Section 2.2.1.

Additionally, in this section, the document analyzes the energy efficiency of the AI/ML models themselves, emphasizing the need to balance computational resource consumption with model performance. Key aspects covered include:

- **Energy Score and Rating Functions**: Metrics to measure and compare the energy efficiency of network components in bits per joule, enabling data-driven optimization.
- **Energy Consumption of AI/ML Models**: Evaluation of trade-offs between model complexity, accuracy, and energy usage, with techniques like dimensionality reduction to minimize overhead.
- **Model Training Energy Measurement**: A framework to quantify the energy cost of training, validating that CPU usage is the dominant factor, and how to measure the energy cost of model training.
- **Model Selection Function**: A proposal for an AI Engine feature to dynamically select the most energy-efficient model that meets accuracy requirements, promoting sustainability in decision-making.

2.2.1 AI Engine Energy score/rating functions

The **Energy Score function** provides an absolute measure of energy efficiency for a component in bits per joule. The AI Engine hosts it as a serverless function to perform the calculation of energy efficiency scores for components in the network. The basic requirement to calculate energy score for a network entity such as a cell or gNB is that the energy consumption and a data volume and/or throughput measures are available. At layers where it is available (e.g., the PDCP layer), the measurement should be net number of retransmissions.

The Energy Score function is a very simple function that performs a calculation based on energy usage and data volume or throughput measurements and returns an 'energy score' as an absolute value in bits per joule. Bits per joule is the standard measurement for EE in telecommunications as defined by the Next Generation Mobile Networks (NGMN) in [13]. Using a standardised measurement with a straightforward formula means that equivalent values can also be calculated for network equipment outside the scope of the AI Engine. To effectively validate the energy savings achieved by the BeGREEN project, the Energy Score must represent an absolute value of EE as opposed to a measure relative to external factors.

The Energy Score function accepts a JSON payload containing two key fields:

- **data_volume**: Represents the volume of data transmitted, measured in megabits (Mbits). This can be a single value or a list of values.
- **energy_consumption**: Represents the energy consumed during the data transmission, measured in watt-hours (Wh). This can also be a single value or a list of values.

The AI Engine function calculates the energy score using the formula, where:

- Data volume is converted from megabits to bits by multiplying by 10^6.
- Energy consumption is converted from watt-hours to joules by multiplying by 3600.

The Energy Score function output returns both individual energy scores (when multiple values are passed in) as well as an overall average score for the aggregated data. The function also includes input validation, error

handling and logging to the AI Engine framework to provide a robust, efficient and scalable solution to EE calculation in BeGREEN.

The **Energy Rating function** is also deployed as a serverless function on the AI Engine. The Energy Rating function is able to provide a relative energy rating (A – E) for any network entity in the BeGREEN ecosystem based on available historical data (including energy consumption and data throughput/volume measurements net of retransmissions at layers where this is available). The most efficient 20% of entities are rated A, the next 20% rated B and so on.

The Energy Rating function also provides a percentile value to give more insight into EE trends in the network. For example, an energy rating will be displayed as 'A 99' or 'C 43,' allowing smaller changes in relative efficiency to be tracked, which would not be possible with an approach limited to the granularity of a quintile. As well as the average value over the defined period, the function also provides the current energy rating, energy rating one hour ago and energy rating one day ago (dependent on data being available). These metrics are key for identification of the network entities that need to be optimized before activating or applying the EE algorithms. The historical ratings and precise percentiles of relative performance provide valuable insight into a network entity's EE performance.

The Energy Rating function calculates relative energy ratings for network entities based on a file or files containing performance data, stored in the AI Engine datalake. The Energy Rating function accepts a JSON payload containing the following fields:

- **target:** list of cells for which to return energy ratings.
- **filelist:** a list of files that contain the data from which to compute energy ratings.
- **bucketname:** the S3 bucket in the AI Engine that contains the file(s) in the filelist.
- **limit:** the number of Result Output Periods (ROPs) to use from the files (the 'limit' most recent ROPs from the file will be used).
- **energyconsumptionfieldname:** the data field in the file(s) that contains the energy consumption information.
- **datavolumefieldnames:** the data fields in the file(s) that contain the data volume.
- **timestampfieldname:** the data field in the file(s) that contains the timestamp.
- **hours:** the number of hours old a pre-existing energy rating calculation can be before it is automatically recalculated.
- **conversionfactor:** adapts to the units used in the data volume fields (e.g. if data volumes are provided in kilobits use conversion factor = 1000 to divide by 1000 to convert to megabits. If provided in megabits use conversion factor = 1).
- **rops_per_hour:** The frequency of data collection used in the files in the filelist; e.g., for 15-minute ROP use 4, for 1-minute ROP use 15.

The function calculates EE ratings (A, B, C, D, E) based on the percentile of the energy score compared to the overall dataset.

The ratings are calculated as follows:

- A: 80-100th percentile
- B: 60-80th percentile
- C: 40-60th percentile
- D: 20-40th percentile

```
[
  {
    "energy_rating":"B 61",
    "energy_score":27.09,
    "current_energy_rating":"D 33",
    "current_energy_score":9.8,
    "ER_1h_ago":"C 46",
    "ES_1h_ago":16.92,
    "ER_1d_ago":"A 87",
    "ES_1d_ago":51.22,
    "ER_1w_ago":"No Data",
    "ES_1w_ago":"No Data"
  }
]
```


Figure 2-3: Energy efficiency in the Intelligence Plane - energy rating function output

- E: 0-20th percentile

An example of the output of the energy rating function is shown Figure 2-3.

This shows the average energy rating and energy score for a cell over a 3-days period together with an energy rating based on its current consumption and volume and snapshots of its metrics one hour ago and one day ago. Section 2.3.2 provides a benchmark for the required resources and the delay associated with serving the energy rating function within the AI Engine.

2.2.2 Energy consumption of BeGREEN functions

Predictive models used in telecommunication network management to anticipate future network conditions and minimize energy consumption can, in fact, become significant contributors to the overall energy usage. These models can consume substantial computational resources for training, inference, and data movement, therefore increasing the energy required to operate these networks. These models are used to determine likely future values for a variable and can leverage vast amounts of data. Both the volume of data that are processed and the type of predictive model used contribute to the resources consumed.

Previous studies [14] have shown that selectively and frugally processing data can minimize the resource consumption of models while retaining their predictive accuracy. In addition to general solution, specific techniques exist for certain model types that can automate the feature selection, such as Recursive Feature Elimination (RFE) [15] for the XGBOOST model. RFE techniques for linear and Support Vector Machine (SVM)-type models are also included in many standard Application Programming Interfaces (APIs). Other model types such as Neural Networks (NNs) or K-Nearest Neighbours can utilize SHapley Additive exPlanation (SHAP) values to identify the least important features to recursively remove from the model.

These techniques involve an initial overhead in computational resources and energy consumption when first applied, as they work by repeatedly fitting the model and removing the least important features one at a time (or in specified batches). However, a model that uses less data will use less computational resources during retraining of the model and during inference. The reduction in data dimensionality can also contribute to reducing other aspects of the model contributing to energy utilization, such as data storage, data movement, memory and I/O.

An evaluation of dimensionality reduction techniques was carried out in BeGREEN D4.2 [2], where the number of input features to a predictive model was limited based on feature importance. Additionally, the effects of various levels of dimensionality reduction were evaluated. The results indicated that significant reductions in CPU load and data size could be achieved at the expense of very small reductions in model accuracy.

Prior literature [16], which relates models to the energy consumption of data centers, identifies the main

components of energy consumption as the CPU, disk, network and memory. To accurately measure the energy consumption of training or inference, each of these factors must be considered.

CPU cycles calculation must consider the Thermal Design Power (TDP) of the processor and its clock speed. For example, for a processor with a stated TDP and clock speed, the calculation of its CPU power consumption (*cpc*) is:

$$cpc = \frac{TDP(W)}{3.6*10^6} * \frac{CPU\_cycles\_used}{CPU\_clock\_speed(Hz)} \quad \text{(EC-1)}$$

To determine the marginal power consumption (*dmpc*) of disk usage, it i dependent on the type of disk, the difference between power consumption of the active and idle states, and the transfer speed of the device. The calculation is as follows:

$$dmpc = active\_pc - idle\_pc$$
$$dpc = \frac{dmpc}{3.6*10^3} * \frac{data\_volume}{data\_transfer\_rate} \quad \text{(EC-2)}$$

For obtaining the marginal network power consumption (*nmpc*) it is necessary to estimate the power consumption of the devices involved in a typical end-to-end transmission of network data, typically a number of Network Interface Cards (NICs), routers and switches. Additionally, the time taken to transmit the data must be calculated, so the bandwidth of the network also factors in the calculation:

$$nmpc = NICs + Routers + Switches$$
$$npc = \frac{nmpc}{3.6*10^3} * \frac{data\_volume}{data\_transfer\_rate} \quad \text{(EC-3)}$$

For calculating the marginal memory power consumption (*mmpc*), it is necessary to estimate the idle and active power consumption of the relevant memory module and its transfer rate. It is also important to include a factor for how many times data is read from memory during a typical ML model training task. This varies greatly between different ML techniques such as regression algorithms and NNs but is typically in the range of 50 to 500 times. To estimate this effect, the data volume is multiplied by 100.

$$mmpc = active\_pc - idle\_pc$$
$$mpc = \frac{data\_volume*100}{data\_transfer\_rate} * \frac{mmpc}{3.6*10^3} \quad \text{(EC-4)}$$

To ensure that the quality of model prediction is maintained, attention must also be paid to quality metrics such as model accuracy. It may be desirable in some cases to use a more efficient model for some predictions than a model with the maximum possible accuracy. There are many scenarios in telecommunication network management where the use of an efficient model that balances speed and accuracy is more beneficial than the use of the more accurate model. An example would be an anomaly detection system to quickly detect and flag potential issues without providing highly accurate diagnostics immediately. Load forecasting during non-peak times when the network has spare capacity can also be done using highly efficient, low accuracy modelling. High frequency short term predictions, such as frequent adjustment of base station power levels, can also perform better when using a faster but less accurate model.

The relative contribution of the components of energy usage was assessed on a range of model training tasks.

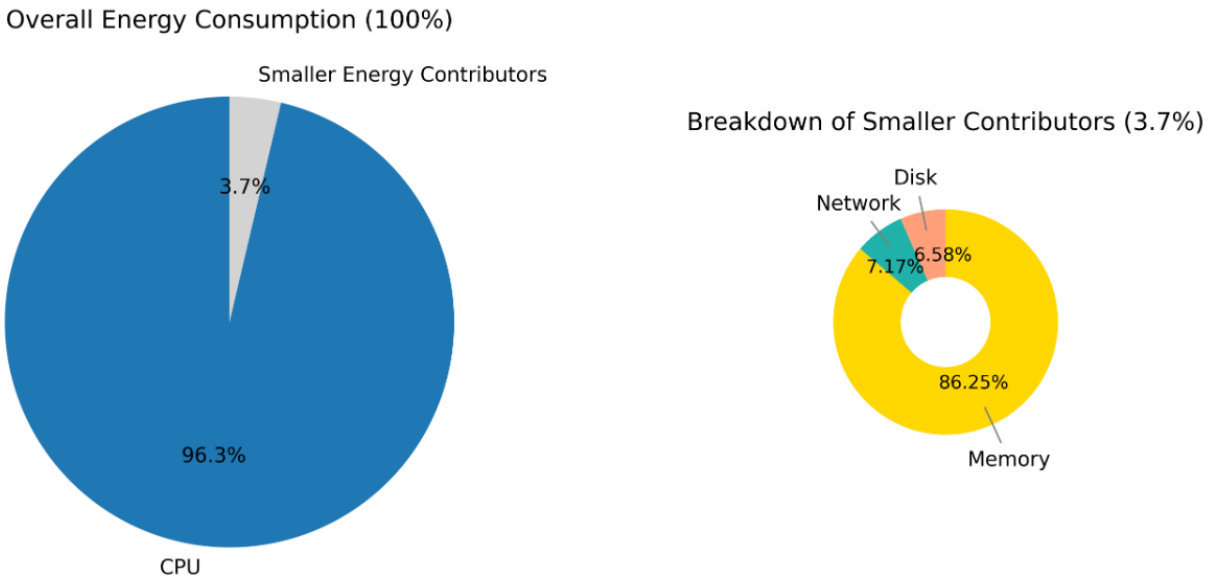


Figure 2-4: Energy efficiency in the Intelligence Plane - Energy Consumption Breakdown

Taking a typical example of a model predicting future energy consumption values for a cell, the total energy consumption was found to come mainly from CPU energy consumption with the other components being much smaller factors, particularly the energy consumption of disk access and network I/O. Disk access could be slightly higher depending on memory paging. This can be seen in Figure 2-4.

The implication of this breakdown is that the energy consumption of training a model will be nearly perfectly linearly correlated with that model's CPU usage. It also indicates that the CPU cycles used and the specification of the processor, would be sufficient to generate a useful estimate of the energy costs associated with training a model, as the contribution of the other factors is extremely small (i.e., < 4%).

2.2.3 Measuring the energy consumption of model training in BeGREEN functions

To validate previous characterization, we measured and estimated the energy consumption of model training carried out by BeGREEN functions. To this end, a list of parameters relating to a typical O-RAN deployment and hardware was assembled as listed in Table 2-1.

Table 2-1: Energy Efficiency in the Intelligence Plane - Example System Parameter List

Parameter	Value
CPU TDP	230W (115W x 2)
CPU Clock Speed	2.6 GHz

As analysed in previous section, since the contribution of factors other than CPU to power consumption were negligible, only the CPU cycles needed to be calculated to obtain a proper estimate of the power consumption of a model training function. The estimated power could then be obtained by adding approximately an 4% overhead to the power consumption value calculated from the CPUs. To demonstrate this, model training functions were tested to find the number of CPU cycles it took to train the model. The amount of data that was passed into the function was also measured. Using these data, an algorithm was developed based on the formulae in the previous section that provided the marginal cost in Watt Hours (Wh) to train each model.

The algorithm is deployed in the Intelligence Plane of BeGREEN as an Energy Measurement serverless function and is accessible for authors of BeGREEN energy saving functions to measure the energy consumption of these functions. To provide this information, the function author must supply the CPU cycles and the data volume used for training the function, together with the system parameters of the execution environment. This allows BeGREEN to verify its net contribution to EE by evaluating the overhead involved

in the energy saving process. The minimum data needed to provide an estimate for a function's energy consumption are the TDP of the processor, the clock speed of the CPU and the number of CPU cycles expended in executing the function. A slightly more accurate measurement can be obtained by applying the formulae in Section 2.2.2 related to disk, network and memory, together with the volume of data that is being passed to the function.

To obtain an accurate measurement of the actual consumption of the energy saving functions in isolation, the measurement of CPU cycles must include all CPU cores used in the computation but exclude any other processes or background services executing on the server. There are many tools that can be used to profile CPU consumption. Intel VTune [17] works across all major platforms and languages, whereas the 'perf' tool [18] is Linux specific but also works across all languages. Other language specific, and platform agnostic, tools are also available such as *psutil* for Python but with adaptations for C++ and Go languages, *OSHI* for Java, *sys-proctable* for Ruby and *System Diagnostics* for C#. Some of these language specific tools vary in their capability to measure CPU cycles across multiple cores. If multiple CPUs with differing clock speeds are used in the execution of the energy saving functions, CPU cycles must be recorded separately for each processor. Then, multiple calls should be made to the Energy Measurement function, and the results must be aggregated to get the total energy consumption of the energy saving function.

To validate this approach, a set of models were trained with 600MB of telecom performance data containing 1400 features on a 32 CPU server with 256GB of memory. Additionally, reduced feature sets of 400, 200, 100, 50, 30, 20 and 10 features as described in [1] were considered. The baseline energy consumption of the server was measured and found to be stable. The models were repeatedly trained for a 30-minute window each, and the statistics for each model were measured using the 'perf' tool in Linux and compared to the power used by the server, which was separately recorded using a power meter.

The results show that the calculated energy used was overestimated by over 50% for the smallest models with the fewest features but converged on around a 10% underestimate for the models trained on larger numbers of features. As the calculated power calculates only the energy consumption of the model training section of the code, the overestimation is due to overheads from the sections of instantiate code outside the main model training function in multiple runs of shorter-lived training functions. The short training time of between 7 and 30 second means that this overhead is a much larger proportion of the total energy consumption than would be the case in a model training scenario, which would by typical for a live network. For models whose training ran for over a minute there was a much lower differential between measured and calculated energy usage.

Models trained in telecommunications functions in live network deployments are typically larger than the samples used here and contain multiple GBs of data and take hours to train so the convergence of calculated and measured energy consumption with larger data sets is the best indicator of the accuracy of this method.

Figure 2-5 shows the energy calculated for each model based on measurements of the usage of CPU taken on the system during execution applied to the system's parameters (as shown in Table 2-1).

Figure 2-6 shows the percentage error of the energy consumption calculation across the different models, with different levels of dimensionality reduction, which were executed for these tests.

Selecting a model with lower dimensionality means that there is less energy consumed during the training of that model. Figure 2-7 indicates how many Watt hours of energy are saved, per training event, by selecting a dimensionally reduced model, based on the measured energy consumption and small network size used in these tests. Energy savings in a live network deployment with more typical models used in telecommunication management functions would be significantly higher.

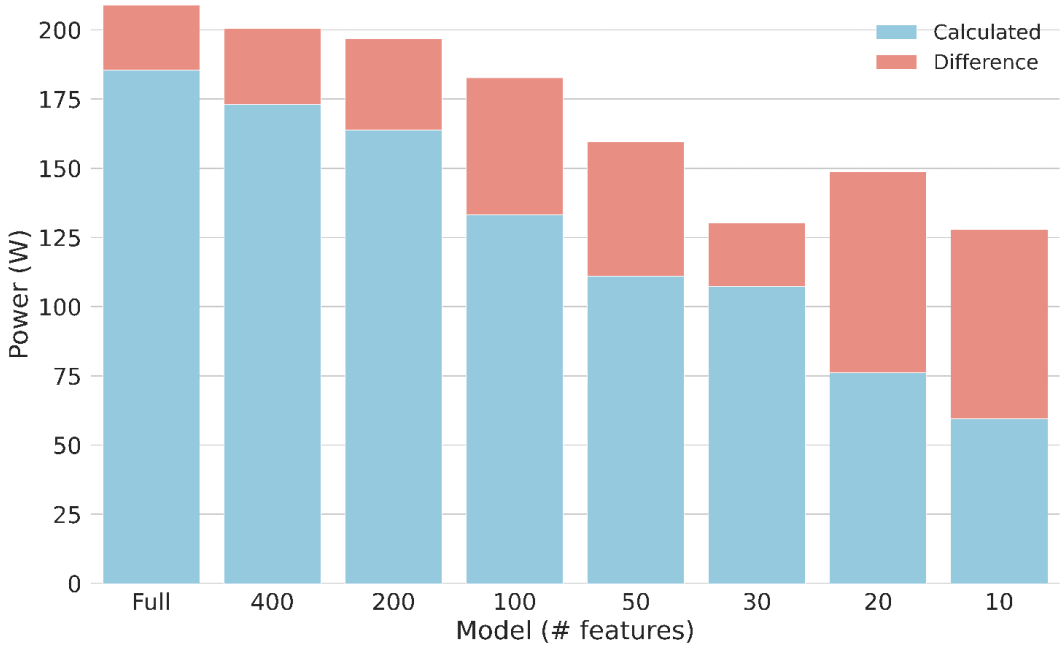


Figure 2-5: Energy efficiency in the Intelligence Plane - Calculated vs Measured power consumption

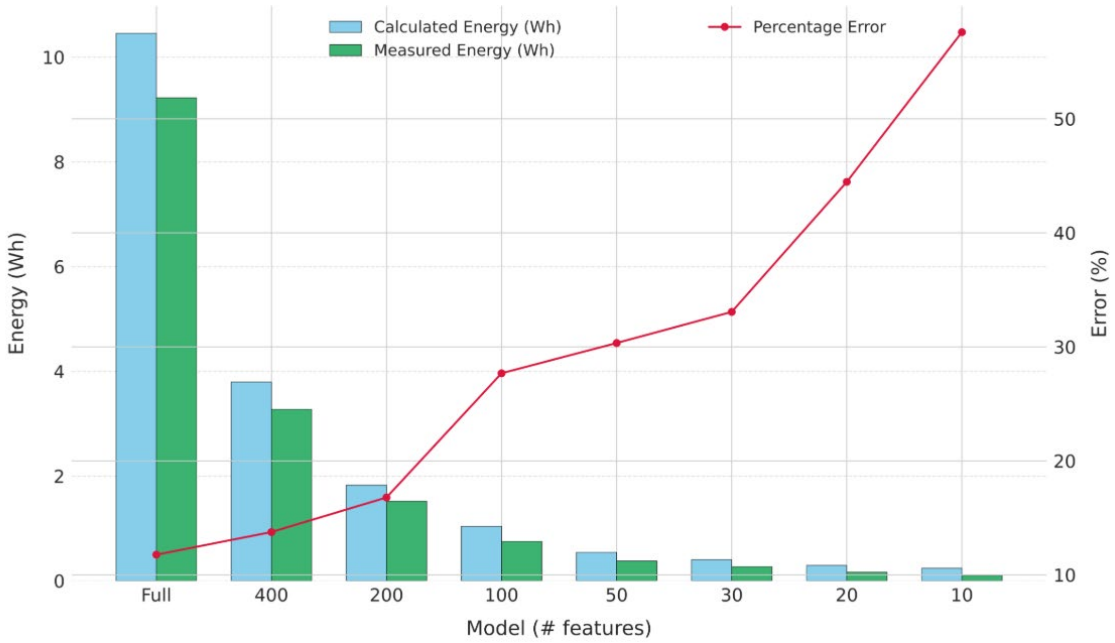


Figure 2-6: Energy efficiency in the Intelligence Plane - Calculated vs Measured Energy with % Error

Due to the disproportionate overheads of very small models, as described above, the error in the calculated energy consumption is high for the smaller models in the test. Figure 2-8 shows the close correlation between CPU utilization measured on the system and the energy consumption measured by the power meter. This graph also indicates the convergence of the Calculated and Measured lines for the larger, more CPU-intensive models.

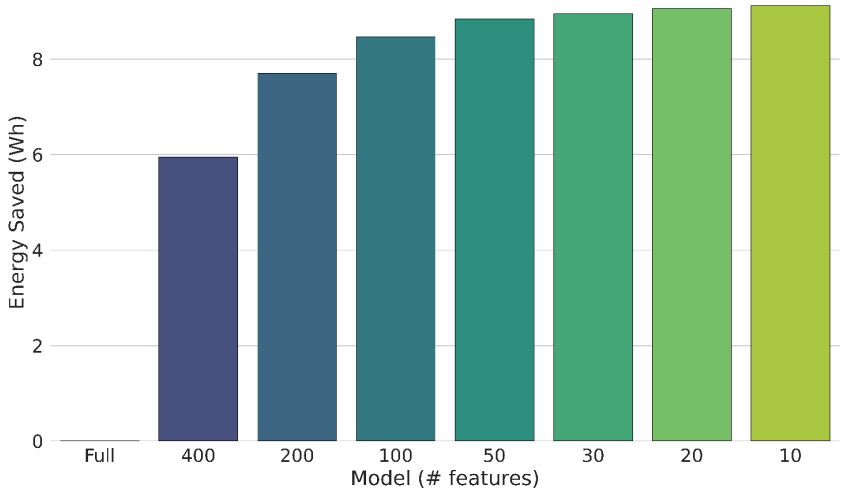


Figure 2-7: energy efficiency in the Intelligence Plane - Energy Savings vs Full Model

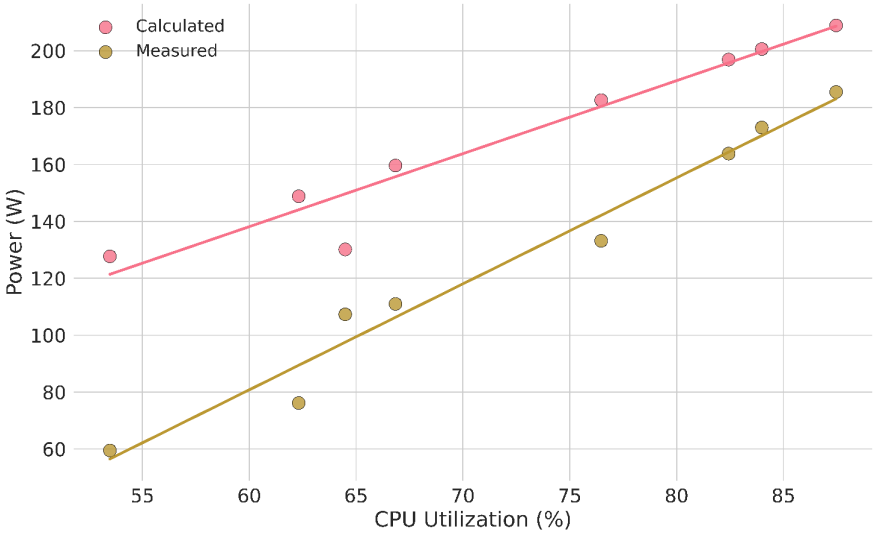


Figure 2-8: Energy efficiency in the Intelligence Plane - Power vs CPU Utilization

2.2.4 Model catalogue and selection function

Selecting the most efficient model to train for a prediction task can save a significant amount of energy over time in a large network deployment with many autonomous optimization applications running constantly and periodically retraining As introduced in section 2.1.1.1, this involves the provision of a Selector function in the AI Engine, designed to automate the selection of the most energy-efficient AI model that meets a specified accuracy requirement. It operates as part of the model deployment and inference pipeline, ensuring optimal resource utilization while maintaining performance standards.

To this end, AIA rApps are no longer tied to a specific model or model serving endpoint in Nuclio; instead, they are associated with an information type, such as 'cell load prediction' or 'cell energy prediction'. When a request is received via R1/ICS from a control rApp for a specific information type and accuracy requirement, the AIA rApp queries the Model Selection function in the AI Engine to determine the most energy-efficient model to fulfil it. The Model Selection function responds with the endpoint of the selected model, which the AIA rApp then uses to start serving the model output. This procedure is triggered each time an rApp requests a model output for the first time, i.e., when a new job subscription is received by the AIA rApp via ICS/R1. The workflow in Figure 2-9 illustrates these operations.

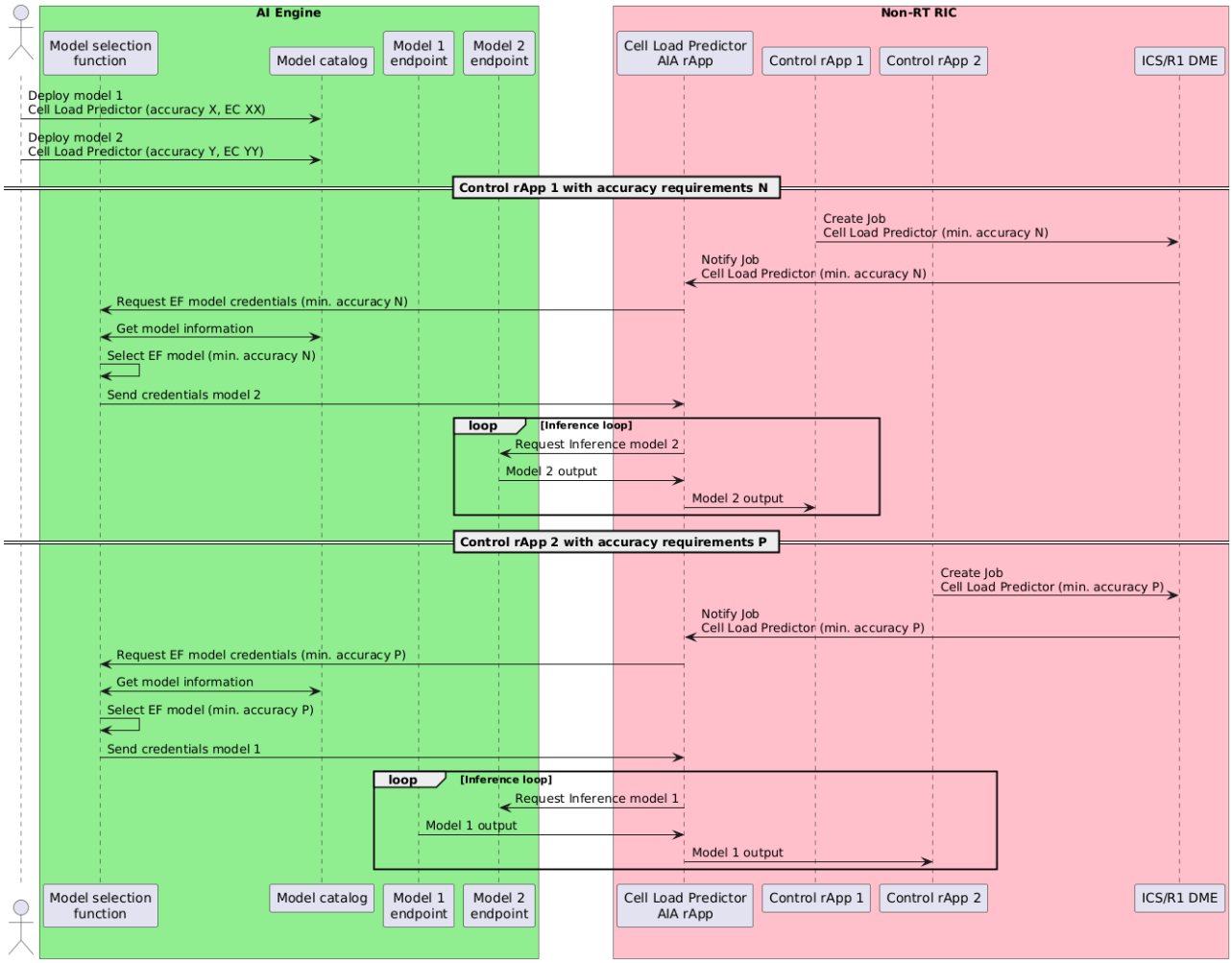


Figure 2-9: Energy efficiency in the Intelligence Plane - model selection architecture

The key functionalities of this procedure are described as follows:

1. Model Selection Based on Accuracy: The function accepts a JSON input containing a *required_accuracy* parameter. It queries the MLRun database to retrieve metadata for all registered models across projects, filtering those that meet or exceed the specified accuracy. The metadata includes metrics such as r2_score (accuracy) and *total_energy_wh* (energy consumption).
2. Energy Efficiency Optimization: From the filtered models, the function selects the one with the lowest energy consumption. This ensures minimal operational overhead while adhering to the accuracy constraints, aligning with sustainability goals in cellular network management.
3. Dynamic Response to Input: The function handles edge cases gracefully, such as invalid JSON input, missing accuracy parameters, or no models meeting the criteria. It provides informative error messages and usage guidelines to facilitate integration with other components (e.g., the AIA rApps).
4. Integration with Non-RT RIC Workflow: The output of the function is a JSON object containing the selected model's name, endpoint, project, accuracy, energy consumption, and energy savings compared to the least efficient viable model that feeds into downstream processes.

By automating model selection with a dual focus on accuracy and EE, this function addresses a key challenge in AI-driven radio resource control: balancing computational performance with sustainability. Its deployment within the Non-RT RIC exemplifies how intelligent orchestration can enhance the scalability and efficiency of cellular networks. The design of the function also highlights the role of metadata (e.g., labels like *r2_score* and *total_energy_wh*) in enabling cross-component interoperability within the O-RAN architecture and

directly contributing to BeGREEN's goal of intelligent, efficient radio control. Demonstration and evaluation of this feature will be included in the PoC1 demo within WP5.

2.3 Validation of the Intelligence Plane

This section details functional validations and benchmarks of the Intelligence Plane implementation done through different PoCs and prototypes. First, we benchmark the AI Engine during model training and serving. Although there is an evident relationship with the hardware capabilities, the analysis provides some relevant insights regarding the trade-off between energy consumption and performance. We then validate the automated generation of datasets in the AI Engine datalake using the R1 interface and producer rApps. Next, we present the integration of the Non-RT and Near-RT RICs. Then, the conflict mitigation performed at the Near-RT RC is detailed, which manages conflictive A1 policies. Finally, the integration of RIS and Intelligence Plane is described in detail and evaluated.

2.3.1 AI Engine Benchmark: training

This section describes the procedure used to characterize the power consumption, the CPU usage and the performance obtained when training ML models in the AI Engine. Concretely, we characterized several multi-output classifiers from which we chose the one which is later used to address the energy-efficient 5G carrier on/off switching in section 3.2. This is aligned with the results obtained in BeGREEN D2.3 [19] (Section 4.8), which concluded that energy-saving strategies should contemplate QoS aspects, since those can impose significant constraints on the switch off opportunities.

The multi-output models are trained to output six binary decisions, related to the defined QoS levels. We compare in total four different models that differ either in the model type or in the number of input features, n. We begin by comparing two different model types (low-complex and mid-complex), which are trained with $n = 15$ input features and three weeks of data, which represent around 400.000 training samples.

The first one is the Ridge classifier[5], which solves a closed-form solution using the Normal Equation or an optimization method like the gradient descent, which typically converges fast as it minimizes a quadratic function. The second one is the Logistic Regression classifier, which we already used in BeGREEN D4.2 [2] as a simple output model to decide 5G carrier on/off switching. This latter uses iterative methods like Newton's method or the Stochastic Gradient Descent (SDG) method, which minimize the log-loss function, a non-quadratic function, and requires a higher number of iterations to converge. From a general perspective, the Ridge Classifier should require less training time, but the Logistic Regression should provide better performance.

After this first comparison, we reduce the number of input features twice and compare two reduced Logistic Regression models to see how this affects the training process. The comparisons we present are based on the following metrics:

- Power consumption: average consumption of the full hardware[6].
- Average CPU usage: average usage of the 12 virtual CPUs the hardware is composed of.
- Training accuracy: measured as the F1 score, the precision and the recall obtained.

To extract the power consumption and CPU usage metrics we use some bash commands that are based on the "*powerstat*" and "*mpstat*" linux commands. Those commands allow us to export the results obtained during the training process into a Comma-Separated Values (CSV) file that we later process using Python.

[5] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RidgeClassifier.html

[6] *Intel NUC10i7FNH, i7-10710U CPU (12 vcores, up to 4.2GHz), 64 GB RAM*

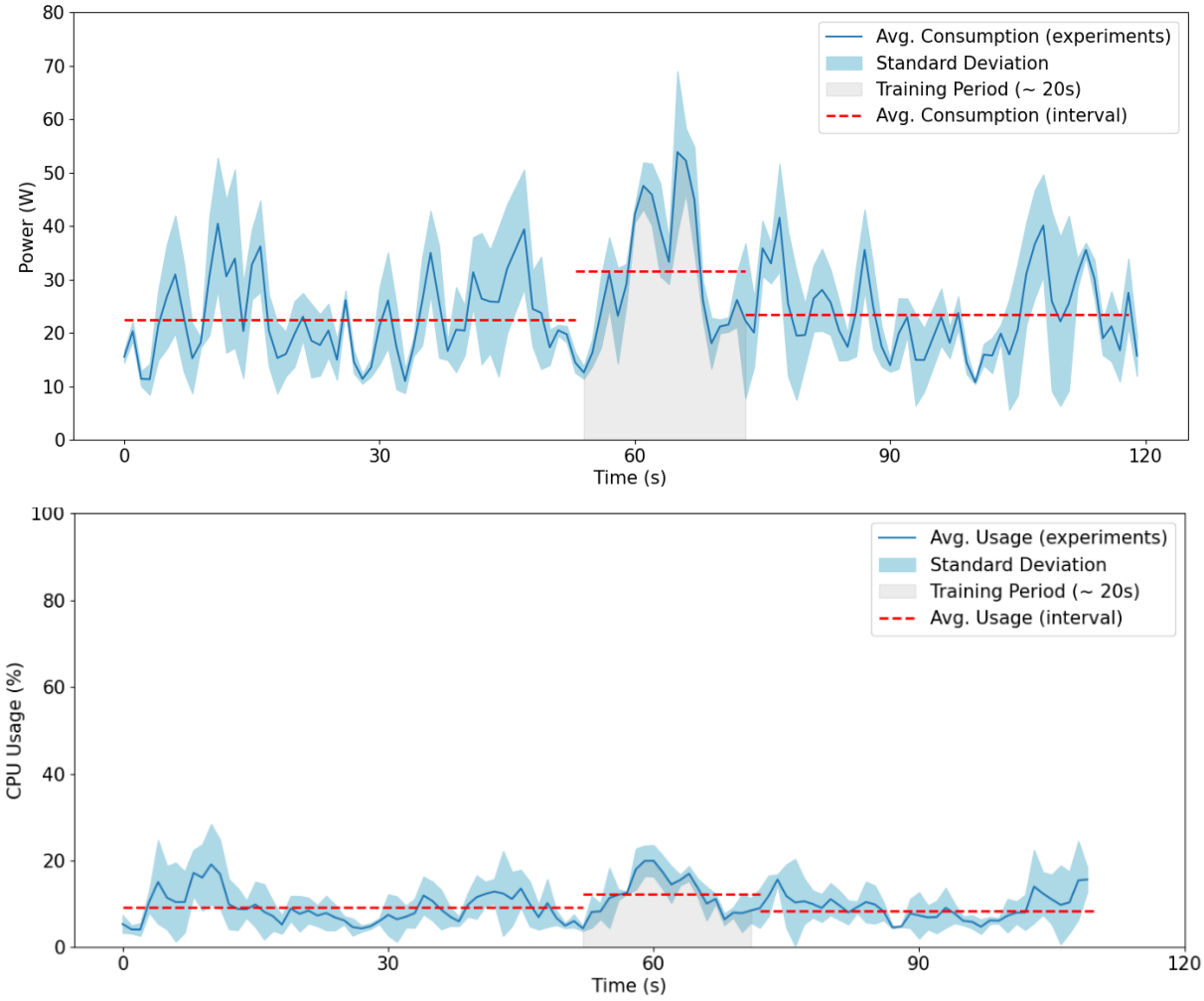


Figure 2-10: AI Engine Benchmark - Ridge Classifier power consumption (top), and CPU usage (bottom)

The ML models are trained on an MLRun instance in the same server. When the training finishes, MLRun provides the model performance metrics regarding accuracy. To provide statistical diversity we repeated all the experiments three times, showing as a result the average and standard deviation obtained across them.

Figure 2-10 and Figure 2-11 show the power consumption (top) and the CPU usage (bottom) for the Ridge and Logistic Regression classifiers, respectively. As expected, the Ridge classifier converges faster (∼ 20s) than the Logistic Regression classifier (∼ 120s), and the impact on the power consumption and CPU usage is much lower. However, Figure 2-12, which captures the performance obtained for all tested models, shows that the Ridge classifier is one point below in the obtained precision, four points below in the obtained recall and three points below in the F1 score.

Since the total energy consumption of the training process for the Logistic Regression classifier was low (50W on average during 120s, i.e., 1.67 Wh) and it was obtaining better results, we decided to perform a feature importance analysis to reduce the number of input features to see if the training time could be reduced without impacting the performance. First, we erased the five less important features from the input set, leaving a total of ten input features, and then we just left the 6 more important features for the carrier on/off switching use case.

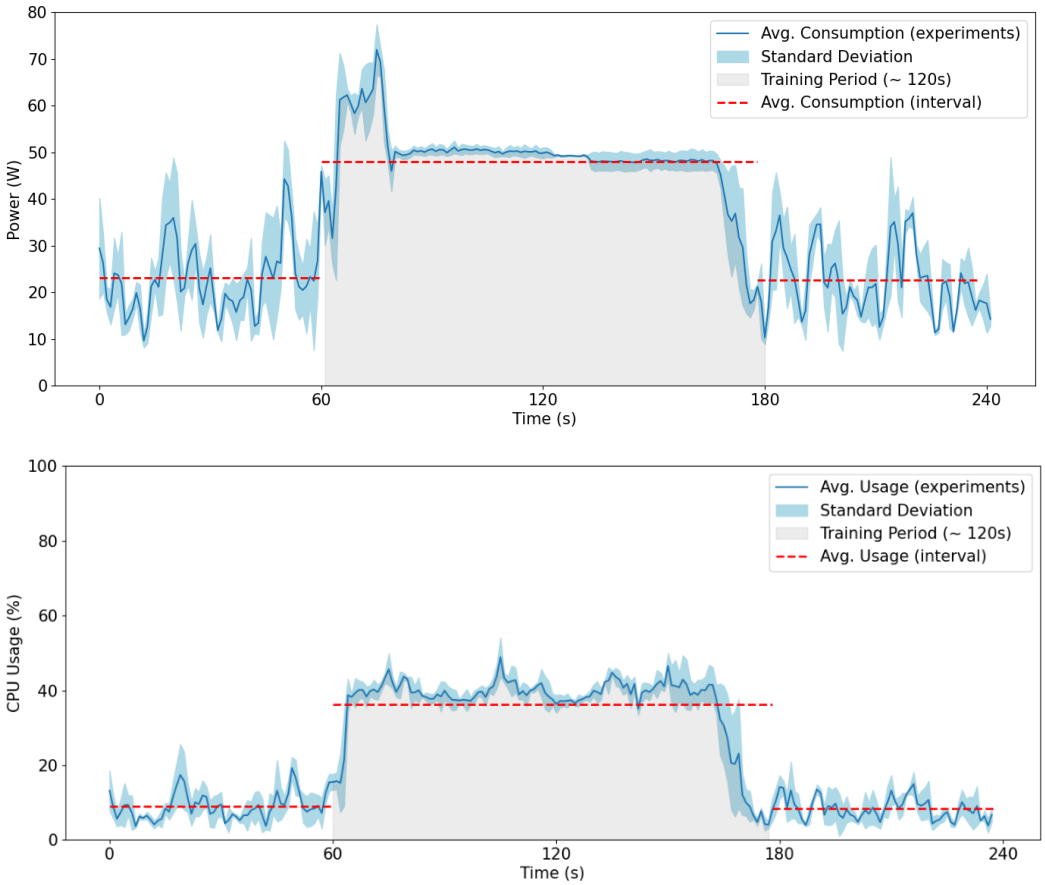


Figure 2-11: AI Engine Benchmark - Logistic Regression Classifier power consumption (top), and CPU usage (bottom)

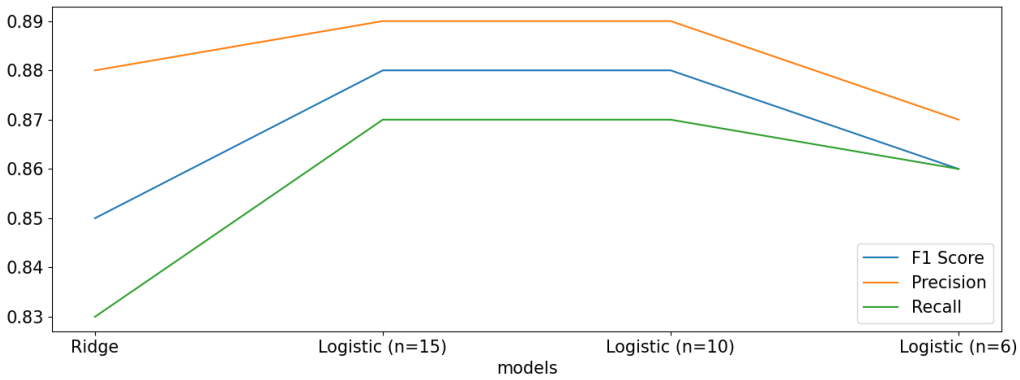


Figure 2-12: AI Engine Benchmark - Performance metrics evaluation for the different tested models

Figure 2-13 and Figure 2-14 show the power consumption results for the Logistic with $n = 10$ and $n = 6$, respectively. Reducing the number of input features decreases the training time and consequently the power consumption and the CPU usage increase. Indeed, Figure 2-12 shows that while reducing from $n = 15$ to $n = 10$ reduces the training time from 120s (1.56Wh) to 90s (1.19 Wh), it has no impact in terms of performance. Reducing to $n = 6$ makes the Logistic Regression classifier to lose some accuracy points but still performs better than the Ridge classifier while reducing training time and power consumption up to 0.53 Wh). Therefore, we decided to use the latter model, i.e., Logistic Regression classifier with $n = 6$, as the one used to address the carrier on/off switching that will be evaluated in section 3.2.

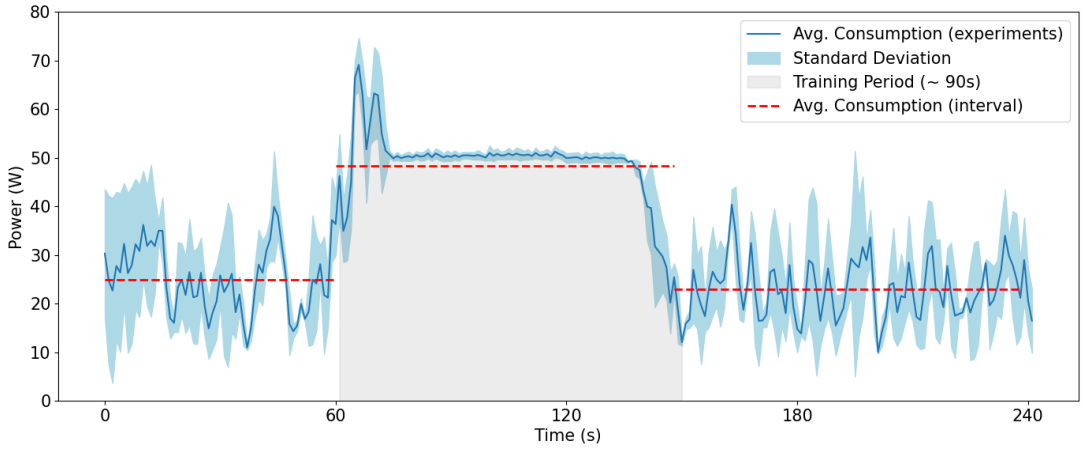


Figure 2-13: AI Engine Benchmark - Logistic Regression Classifier power consumption (n=10)

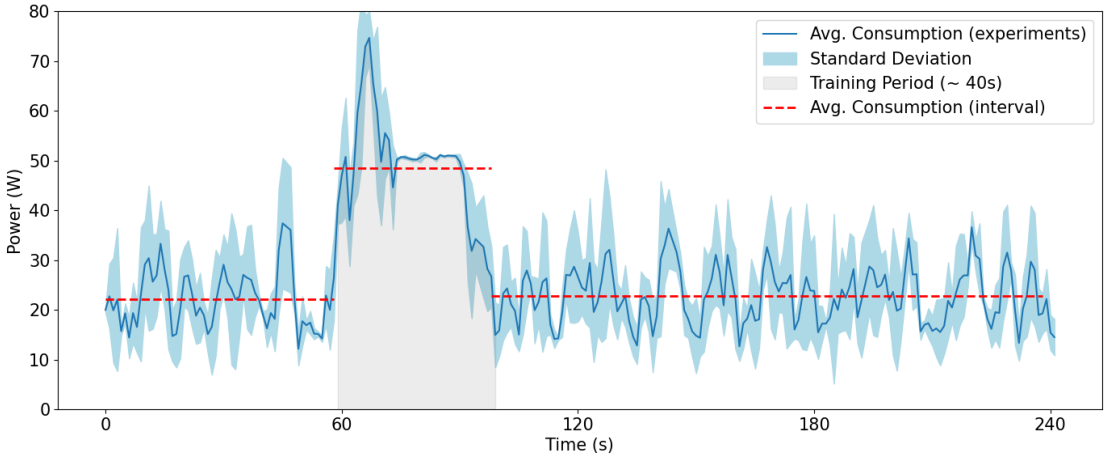


Figure 2-14: AI Engine Benchmark - Logistic Regression classifier power consumption (n=6)

Finally, with the selected model, we performed a further test to evaluate whether reducing the CPU frequency during training could provide any benefit in terms of power consumed. Figure 2-15 and Figure 2-16 show the comparison between 3 GHz (high frequency) and 1 GHz (low frequency). Note that previous tests were performed using dynamic frequency setting by the powersave CPU governor.

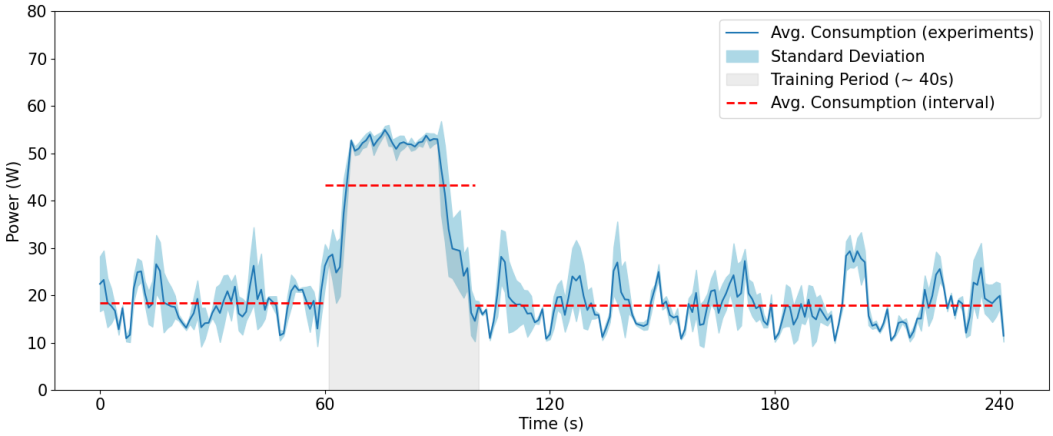


Figure 2-15: AI Engine Benchmark - Logistic Regression Classifier with n=6 and CPUs at high frequency

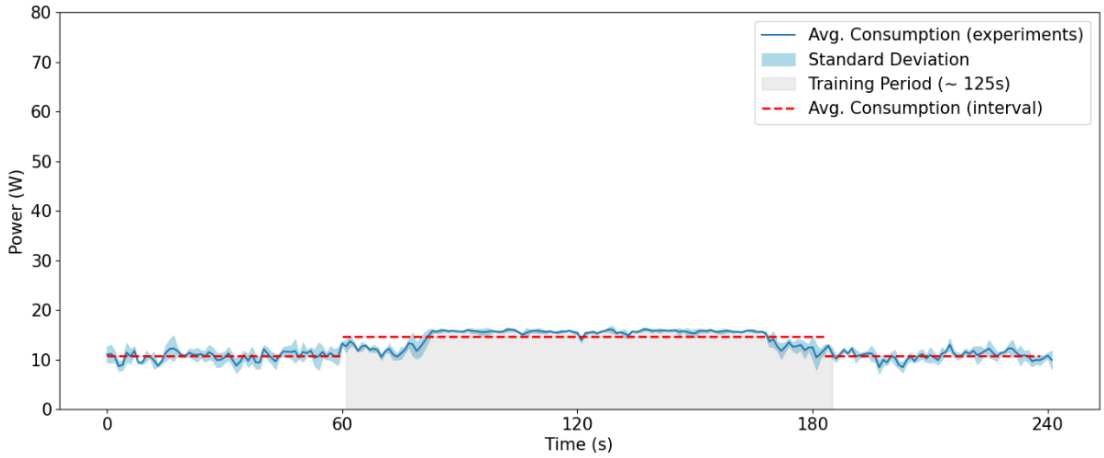


Figure 2-16: AI Engine Benchmark - Logistic Regression Classifier with n=6 and CPUs at low frequency

As expected, we observe that power consumption at low frequency is much lower but extends for a greater period of time. Therefore, we computed the energy consumption (Wh) according to Equation (2-1) for both cases:

$$E_{cons} = (P_{avg}(W) \cdot T_{training}(s)) \div 3600s \quad (2\text{-}1)$$

- High frequency: $E_{cons} = (44W \cdot 40s) \div 3600s \simeq 0.51\,Wh$
- Low frequency: $E_{cons} = (14W \cdot 125s) \div 3600s \simeq 0.48\,Wh$

Therefore, in terms of total energy consumption there seems to be no benefit from training at low frequency. However, comparing the power consumed during the training to the baseline consumption of the hardware leads to higher differences in high frequency case. Considering this delta, the results show that training at a high frequency doubles the energy consumption compared to the low frequency case. Therefore, when compared to the baseline consumption of the equipment, low CPU frequency is better in terms of energy expenditure. This suggest that CPU frequency policies which consider server and training workloads, like the proposed in section 3.4 for the UPF, could benefit energy savings.

- High frequency: $\Delta E_{cons} = (26W \cdot 40s) \div 3600s \simeq 0.29\,Wh$
- Low frequency: $\Delta E_{cons} = (4W \cdot 125s) \div 3600s \simeq 0.14\,Wh$

2.3.2 AI Engine Benchmark: serving

The AI Engine and the Non-RT RIC must ensure that data and ML model outputs are exposed with enough granularity to implement Non-RT control loops. As was discussed in BeGREEN D4.2 [2], the functions deployed in the AI Engine can lead to bottlenecks according to the workload in terms of consumer rApps or control loop intervals. However, note that Nuclio allows serverless deployments on Kubernetes, what will ensure scalability in production environments consisting of distributed servers. Nevertheless, the provided benchmark in a single server[7] highlights some relevant trade-offs of the developed functions.

We first repeated some of the experiments that were provided in BeGREEN D4.2 [2], comparing the performance of two predictors (energy and load per cell) based on the XGBOOST model. We calculated the excess delay as the difference between the required period and the measured period, according to the number of consumers, of jobs and the period.

[7] *Intel NUC10i7FNH, i7-10710U CPU (12 vcores, up to 4.2GHz), 64 GB RAM*

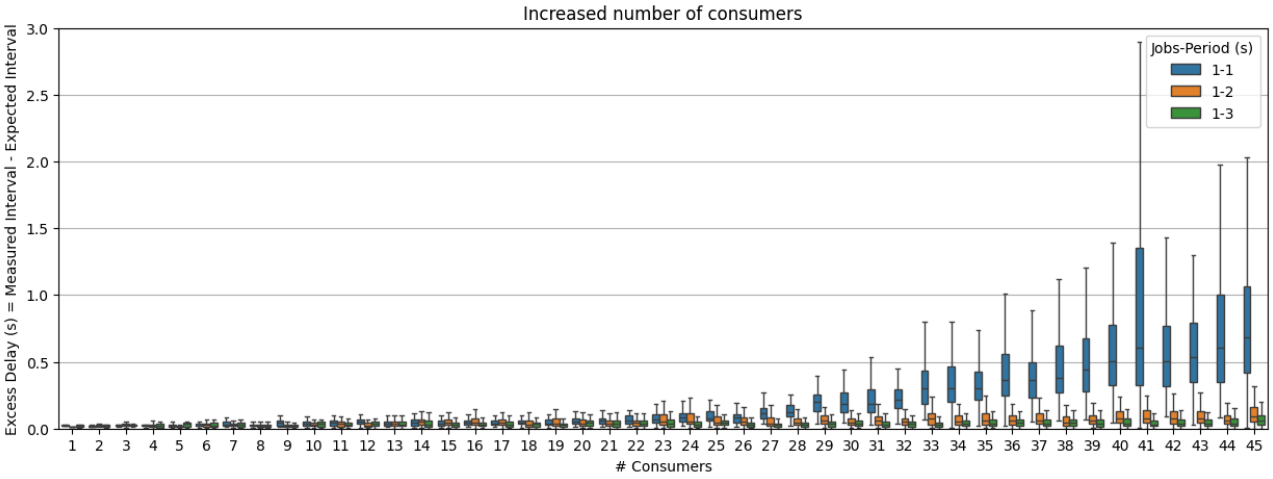


Figure 2-17: AI Engine Benchmark - Jobs latency with an increasing number of consumers (1 job)

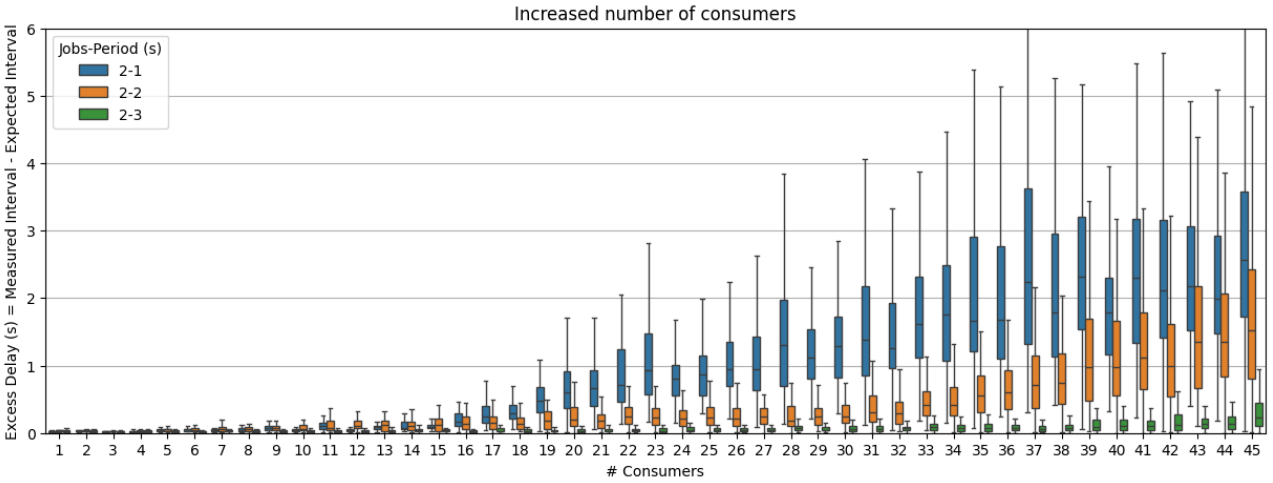


Figure 2-18: AI Engine Benchmark - Jobs latency with an increasing number of consumers (2 jobs)

As depicted in Figure 2-17 and Figure 2-18, the latency increases with the workload of the inference services in terms of request per second: i.e., more consumers and/or jobs, and less interval between requests. In BeGREEN D4.2 [2], we analysed this increase and found that it was caused by the model inference procedures. For instance, when serving simpler functions, such as the energy score, we did not observe this trade-off.

In the case of the energy rating function, since it requires to gather historical data from the datalake, it can lead to longer latencies which may impact excess delay. We characterized also its performance in the AI Engine according to the number of consumers. In the first experiment, the number of cells was fixed to 5 and the job period to 10 seconds, while we considered different ROP limits. Figure 2-19 shows how the excess delay significantly increases with the number of consumers, especially with high limit values since the number of considered ROPs is higher. Recall that the considered dataset contains data with a granularity of 15 minutes; therefore, the 600 limit corresponds to almost one week of data.

We also evaluated the impact of increasing the number of cells with a fixed limit ROP (240), number of consumers (10) and job period (10 seconds). As depicted in Figure 2-20, while latency seems to increase with the number of cells, the increment is marginal compared to the effect of increasing the limit parameter.

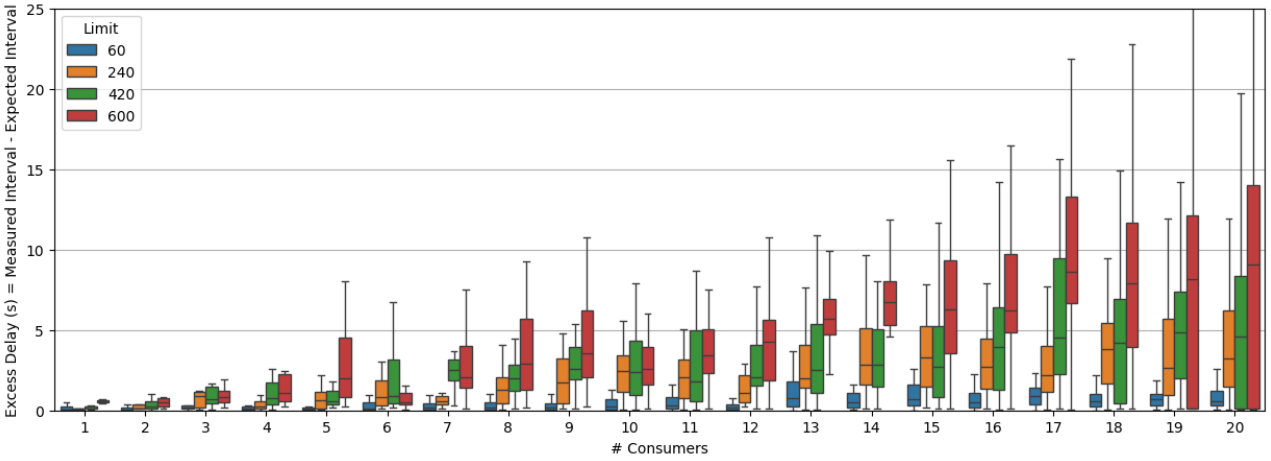


Figure 2-19: AI Engine Benchmark: energy rating latency with an increasing number of consumers and variable limit

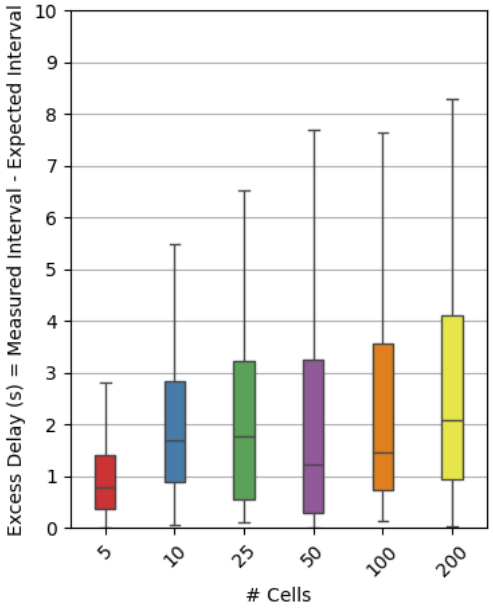


Figure 2-20: AI Engine Benchmark - Energy rating latency with an increasing number of cells

We next characterize the impact of model serving on the resource consumption of the AI Engine host. This helps to fully analyse the implications of the utilization of AI to enhance energy efficiency and to understand the experimented bottlenecks in the server. We fixed the job period to 1 second and the number of jobs per consumer to 2 (load and energy predictors), and characterized the delay, the power consumption and the CPU utilisation of the AI Engine according to the number of consumer rApps. Figure 2-21 illustrates the results with the default CPU governor (i.e., powersave). Recall that the AI Engine server begins to act as a bottleneck when implementing 1-second control loops with 20 consumers, as CPU usage reaches its limit. Consequently, with 40 consumers, delays further increase. The power consumption of the server also stabilizes near its TDP limits.

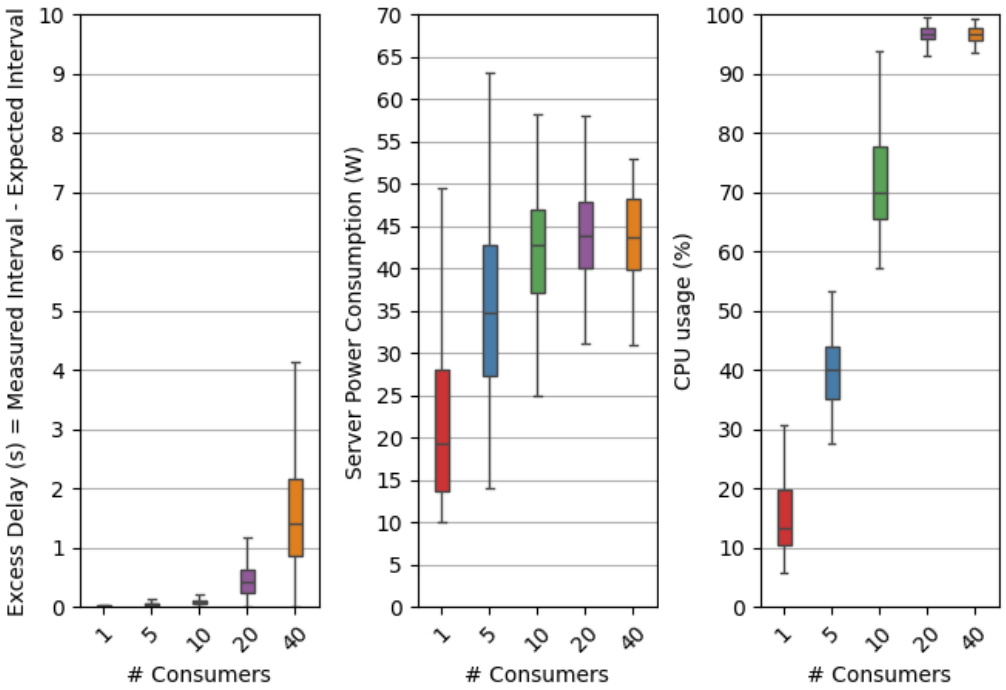


Figure 2-21: AI Engine Benchmark – Resource utilisation during inference with an increasing number of consumers (variable CPU frequency)

Figure 2-22 shows the equivalent results when fixing the CPU frequency to a low value (i.e., 1GHz). In this case, workload reaches server's CPU utilisation limit after 5 consumers, leading to high delays in the case of 10, 20 and 40 consumers. On the contrary, the power consumption remains stable and very low due to the low CPU frequency. Conversely, Figure 2-23 illustrates results when using a fixed higher CPU frequency (i.e., 3 GHz). Notably, delay performance is similar to the CPU free mode case depicted in Figure 2-21, but power consumption gets decreased by avoiding CPU frequency peaks (up to 4.7 GHz in this server). As described in section 2.3.1, these results suggest potential benefits of CPU frequency management policies for MLOps, following a similar approach to that outlined in section 3.4 for servers hosting the UPF. Additionally, in production deployments, the serverless Kubernetes-based architecture of the AI Engine could be leveraged to deploy multiple function replicas on servers operating at lower CPU frequencies, potentially increasing the EE of the system.

Finally, we also analysed the impact of the number of input features on the performance of the Logistic Regression classifier described in section 2.3.1. Notably, as shown in Figure 2-24, and compared to previous results with *xgboost*-based predictors and the energy rating, the latency performance remains very low independently of the number of consumers. Accordingly, same occurs with power consumption and CPU usage, confirming the absence of bottlenecks while serving these models. Results also indicate that the number of input features, i.e., the model complexity, does not seem to impact the serving time or the consumption of resources during inference. This confirms that model selection function, introduced in section 2.2.4, should mainly focus on the trade-off between the resources consumed during training operations and the accuracy metrics of the models.

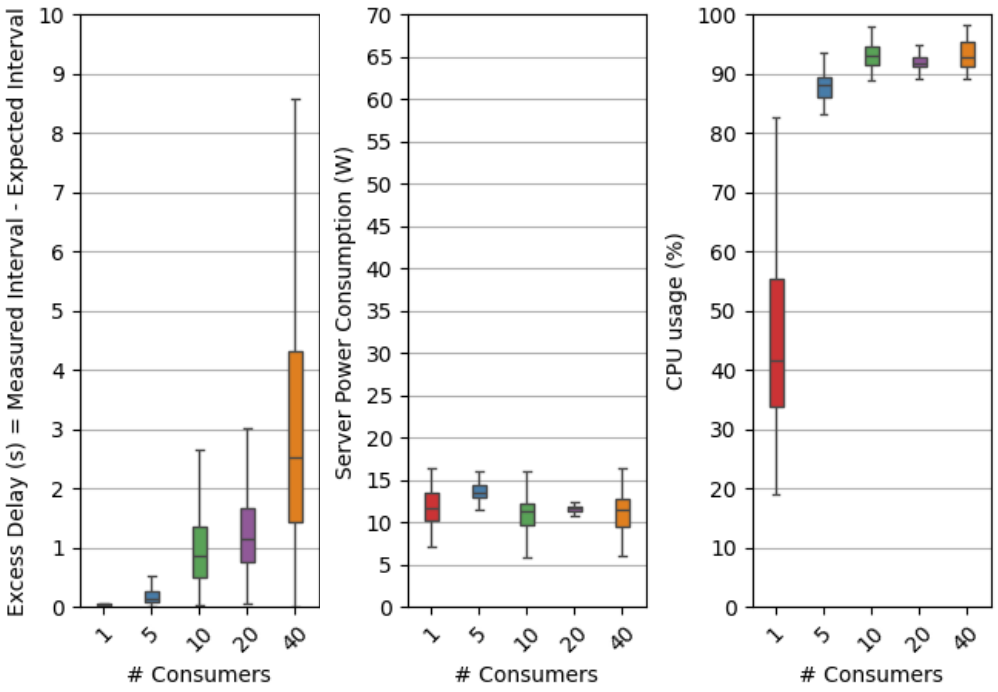


Figure 2-22: AI Engine Benchmark – Resource utilisation during inference with an increasing number of consumers (low CPU frequency)

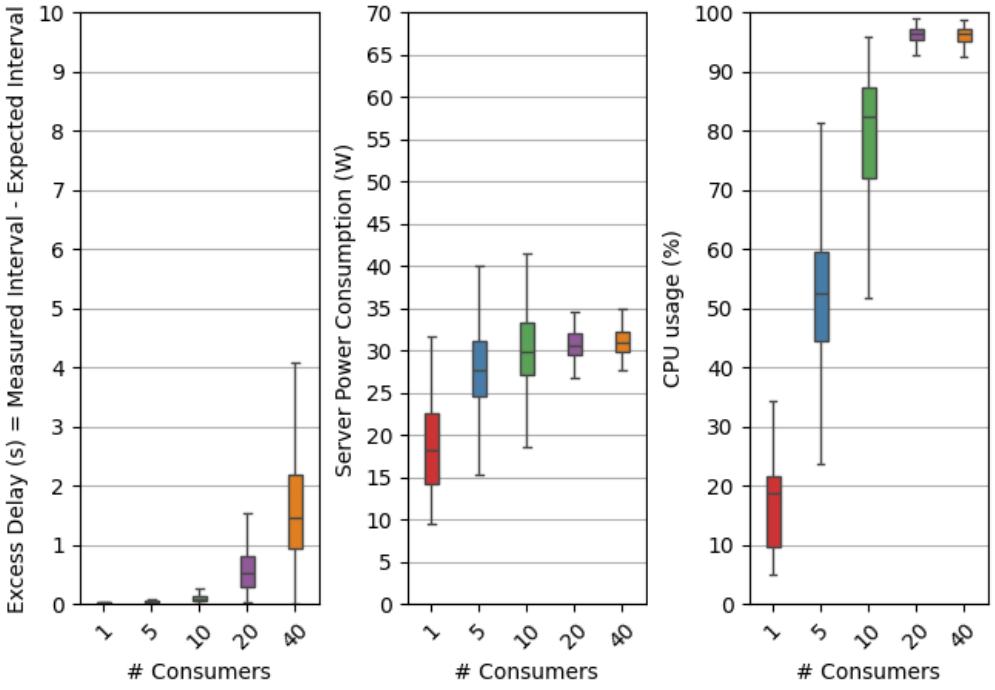


Figure 2-23: AI Engine Benchmark – Resource utilisation during inference with an increasing number of consumers (high CPU frequency)

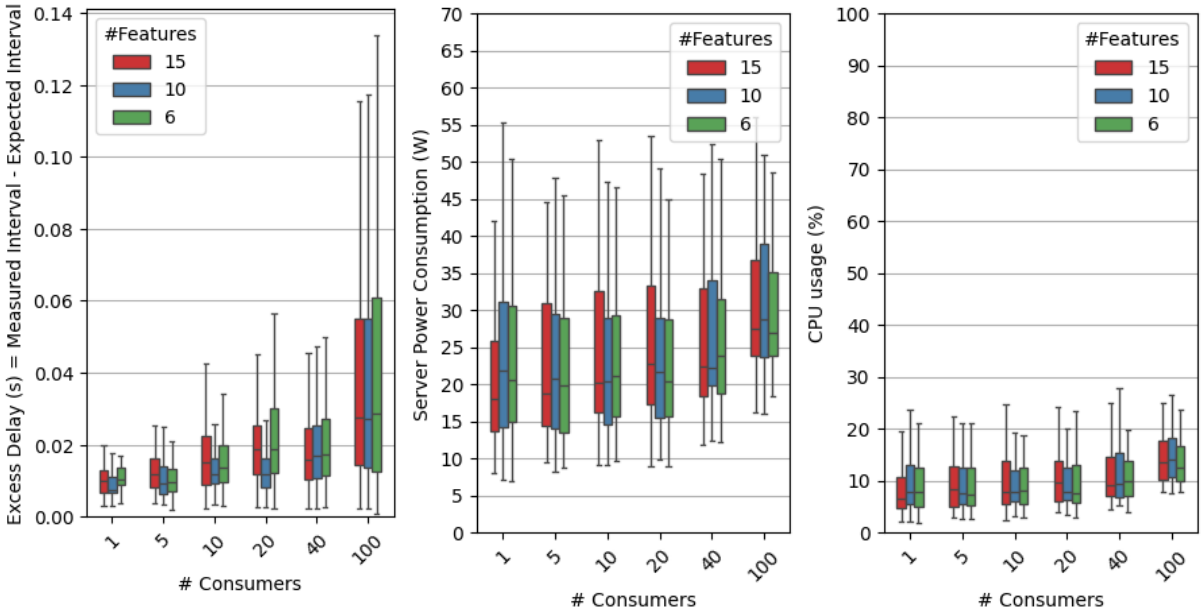


Figure 2-24: AI Engine Benchmark – Performance of classifiers during inference according to the number of consumers and input features

2.3.3 Dataset generation

To automate the generation of datasets according to exposed network KPIs, we have developed a producer rApp which enables KPI collection, processing and uploading to the AI Engine datalake, which is based on Minio[8]. The creation of dataset jobs leverages the producer-consumer paradigm enabled by the R1 interface, which allows several consumers to initiate parallel dataset jobs with different configurations for data retrieval and storage. Dataset jobs can be generated manually through the Non-RT RIC API (e.g., by AI developers) or automatically by rApps. For instance, AIA rApps may initiate a dataset job for enabling retraining procedures, for instance after monitoring performance degradation or according to a policy aiming at using updated data.

Figure 2-25 shows the definition of the *dataset* information type in R1/ICS, which is then served by the "*Dataset Producer rApp*". As illustrated by the parameters, the job definition allows to configure the duration of the job and the period of the data, which is used when generating the secondary jobs required to obtain the data. These jobs are defined in the *data_description* field, providing a list of information types and the parameters required to create them according to their definition in R1/ICS.

The gathered data is stored in CSV format with timestamps. CSV files of maximum size *chunk_size* rows are generated and uploaded during data obtention to avoid the generation of huge files which would complicate the storage and uploading procedure. Bucket and file prefix names are used to identify the dataset. The latter is combined with a timestamp to identify the different data segments or CSVs of a campaign.

Figure 2-26 illustrates an exemplar workflow, involving the rApp exposing KPMs from the near-RT RIC. As shown in the workflow, the dataset rApp manages the life cycle of secondary jobs through R1/ICS.

[8] https://min.io/

Curl

```
curl -X 'GET' \
  'http://192.168.40.51:30000/info-type?type_id=dataset' \
  -H 'accept: application/json'
```

Request URL

```
http://192.168.40.51:30000/info-type?type_id=dataset
```

Server response

Code | Details

200

Response body


```
{
  "info_job_data_schema": {
    "job_definition": {
      "period_secs": "period in seconds of data retrieval (int)",
      "limit_secs": "duration of the job in seconds (int)",
      "chunk_size": "maximum size in rows of the uploaded CSVs",
      "data_description": [
        {
          "info_type": "Information Type defined in R1/ICS",
          "job_description": "Parameters requiered by the   information type according to its ICS definition to generate a job"
        }
      ],
      "minio_credentials": {
        "url": "Minio url (string)",
        "access_key": "Access key (string)",
        "secret_key": "Secret key (string)",
        "bucket_name": "Minio bucket name. Existent or new. (string)",
        "file_prefix": "Prefix (including folder) to be used to generate CSV file names"
      }
    },
    "job_data": {
      "result_list": "Notifies finalization of the dataset job"
    }
  },
  "info_type_information": {
    "description": "Minio-based dataset generation according to metrics being exposed in R1/ICS"
  }
}
```


Figure 2-25: Dataset generation service - Dataset information type

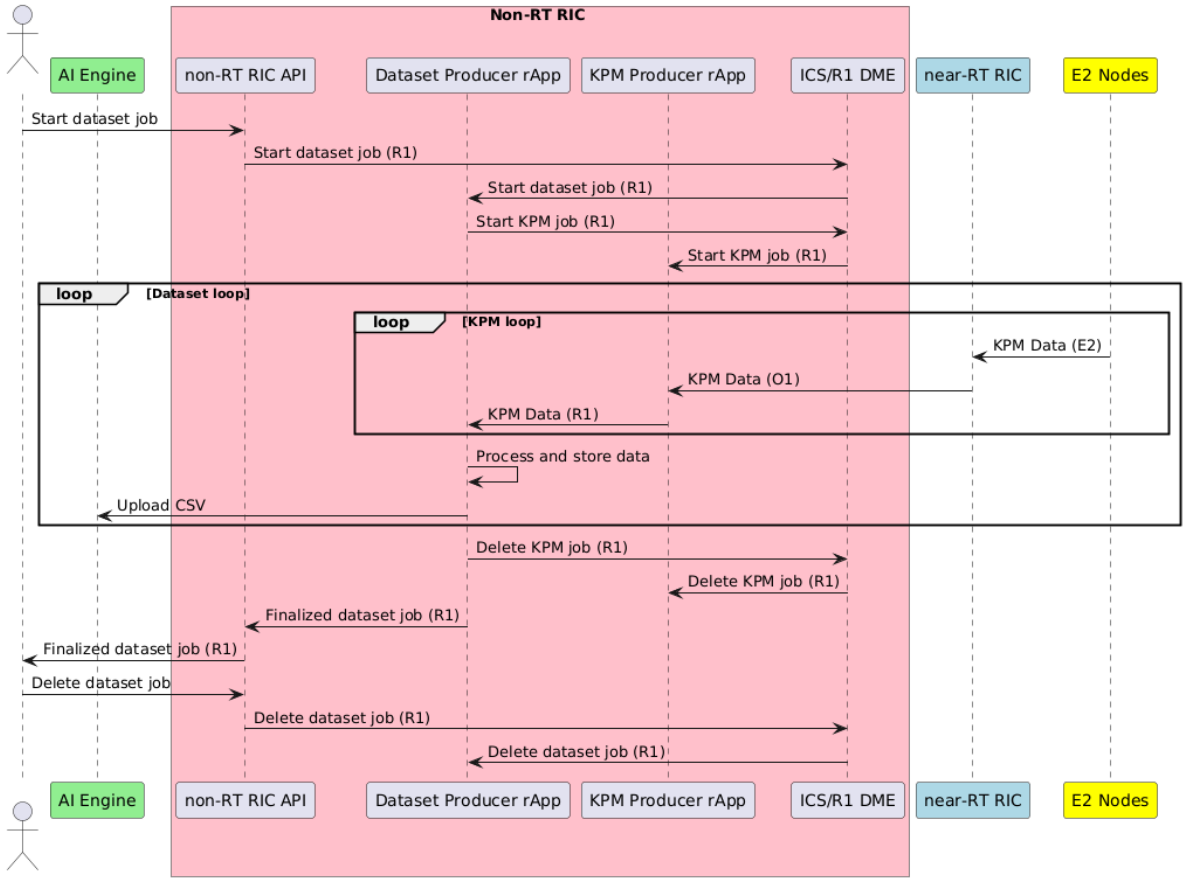


Figure 2-26: Dataset generation service - Exemplar workflow

Curl

```
curl -X 'GET' \
  'http://192.168.40.51:30000/jobs' \
  -H 'accept: application/json'
```

Request URL

```
http://192.168.40.51:30000/jobs
```

Server response

Code	Details
200	Response body

```
[
  "NONRTRIC-dataset-1743086643",
  "dataset-producer-rapp+NONRTRIC-dataset-1743086643+kafka_kpm_cell_drax"
]
```

Figure 2-27: Dataset generation service - Jobs generated in the ICS/R1 after requesting the dataset job

Curl

```
curl -X 'GET' \
  'http://192.168.40.51:30000/jobs?jobid=NONRTRIC-dataset-1743086643' \
  -H 'accept: application/json'
```

Request URL

```
http://192.168.40.51:30000/jobs?jobid=NONRTRIC-dataset-1743086643
```

Server response

Code	Details
200	Response body

```
{
  "info_type_id": "dataset",
  "job_owner": "NONRTRIC",
  "job_definition": {
    "chunk_size": 720,
    "data_description": [
      {
        "info_type": "kafka_kpm_cell_drax",
        "job_description": {
          "metrics": [
            "PEE.AvgPower",
            "PEE.Energy",
            "RRU.PrbTotDl",
            "RRU.PrbTotUl",
            "RRU.PrbAvailDl",
            "DRB.UEThpDl",
            "DRB.UEThpUl",
            "DRB.MeanActiveUe"
          ],
          "period": 5,
          "cell_ids": [
            "4886716420-001-01",
            "4886716421-001-01",
            "4886716422-001-01",
            "4886716417-001-01",
            "4886716418-001-01",
            "4886716419-001-01"
          ]
        }
      }
    ],
    "limit_secs": 7200,
    "minio_credentials": {
      "url": "192.168.40.89:30080",
      "access_key": "W9mDkvOriIVILXk3",
      "secret_key": "ZACcpOk1wX88WxnTE66V7cVVwdhpDwLT",
      "bucket_name": "dataset-ric-march25",
      "prefix_name": "test4/kpmdrax"
    },
    "period_secs": 5
  },
  "job_result_uri": "http://192.168.40.51:30000/datadelivery",
  "status_notification_uri": "http://192.168.40.51:30000/status"
}
```

Figure 2-28: Dataset generation service - Dataset job

This workflow is initiated by creating a job through the non-RT RIC API, which contains the parameters depicted in Figure 2-28. As mentioned, the *data_description* field contains the information needed to create the secondary job to gather the KPMs, of type *kafka_kpm_cell_drax*, provided by the specific rApp. Figure 2-27 shows both active jobs in the R1/ICS DME service.

Figure 2-29 and Figure 2-30 depict the generated bucket and the uploaded CSV files in the Minio GUI, while Figure 2-31 shows an example of the content of a CSV with the gathered metrics and correspondent timestamps and identifiers.

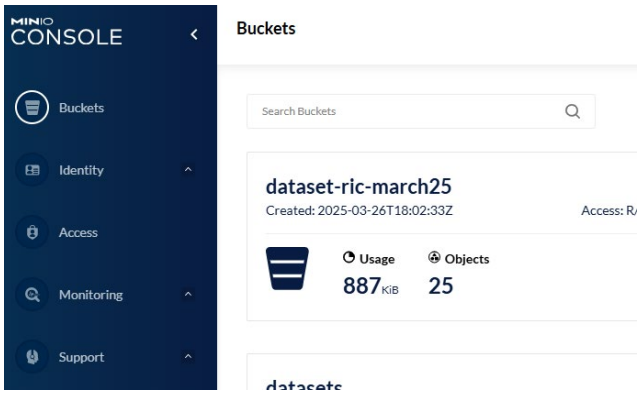


Figure 2-29: Dataset generation service - Minio buckets

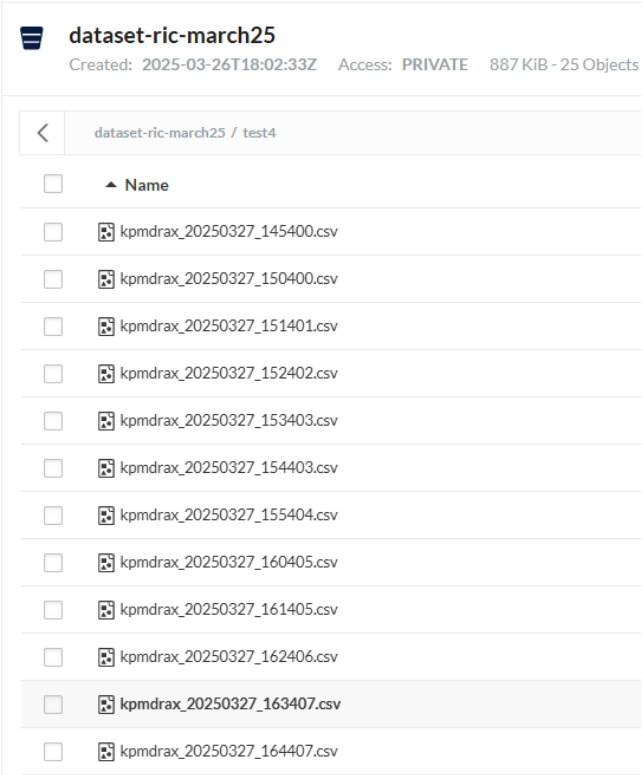


Figure 2-30: Dataset generation service - Minio CSVs

	A	B	C	D	E	H	J
1	timestamp	id	kafka_kpm_cell_drax-PEE.AvgPower	kafka_kpm_cell_drax-PEE.Energy	kafka_kpm_cell_drax-RRU.PrbTotDl	kafka_kpm_cell_drax-DRB.UEThpDl	kafka_kpm_cell_drax-DRB.MeanActiveUe
2	27/03/2025 16:34:12	4886716420-001-01	44.0075	0.324309	7	2545.6	4
3	27/03/2025 16:34:12	4886716421-001-01	58.2665	0.451922	12	4620.8	7
4	27/03/2025 16:34:12	4886716422-001-01	35.2775	0.325919	5	544.8	1
5	27/03/2025 16:34:12	4886716417-001-01	110.064	0.646765	30	4457.6	7
6	27/03/2025 16:34:12	4886716418-001-01	29.4575	0.180224	2	672	1
7	27/03/2025 16:34:12	4886716419-001-01	23.6375	0.150426	0	0	0
8	27/03/2025 16:34:17	4886716420-001-01	49.5365	0.324372	9	2418.4	4
9	27/03/2025 16:34:17	4886716421-001-01	64.0865	0.45202	15	4967.2	7
10	27/03/2025 16:34:17	4886716422-001-01	64.0865	0.325982	6	546.4	1
11	27/03/2025 16:34:17	4886716417-001-01	135.964	0.646949	33	4228	7
12	27/03/2025 16:34:17	4886716418-001-01	29.4575	0.180265	2	576.8	1
13	27/03/2025 16:34:17	4886716419-001-01	23.6375	0.150459	0	0	0
14	27/03/2025 16:34:22	4886716420-001-01	66.9965	0.324447	16	2179.2	4
15	27/03/2025 16:34:22	4886716421-001-01	89.9855	0.452121	17	4500	7
16	27/03/2025 16:34:22	4886716422-001-01	44.0075	0.326042	14	498.4	1
17	27/03/2025 16:34:22	4886716417-001-01	161.572	0.647135	41	4108	7
18	27/03/2025 16:34:22	4886716418-001-01	29.4575	0.180306	2	519.2	1
19	27/03/2025 16:34:22	4886716419-001-01	23.6375	0.150492	0	0	0
20	27/03/2025 16:34:27	4886716420-001-01	66.9965	0.324564	14	2153.6	4

Figure 2-31: Dataset generation service - Example of generated CSV

Once finalized (according to the parameter *limit_secs*), the dataset Producer rApp send a notification to the consumer (the non-RT RIC API in the example), stops the secondary jobs (KPM in the example) and waits for the job deletion by the consumer.

This service is being exploited to generate the required datasets for PoC1 demonstration, which will be provided in BeGREEN D4.4.

2.3.4 RICs integration

The baseline functionalities of BeGREEN's Non-RT RIC, as summarized in Figure 2-32, were presented in BeGREEN D4.2 [2]: rApp life cycle management (LCM), Data Management and Exposure (DME) through R1/ICS, rApps as data produces (including Assist rApps exposing the AI Engine) and rApp as data consumers (including RAN control rApps). In this deliverable it's reported the integration of the Non-RT and Near-RT RICs, which required two main functions: Near-RT RIC registration and A1 policy management.

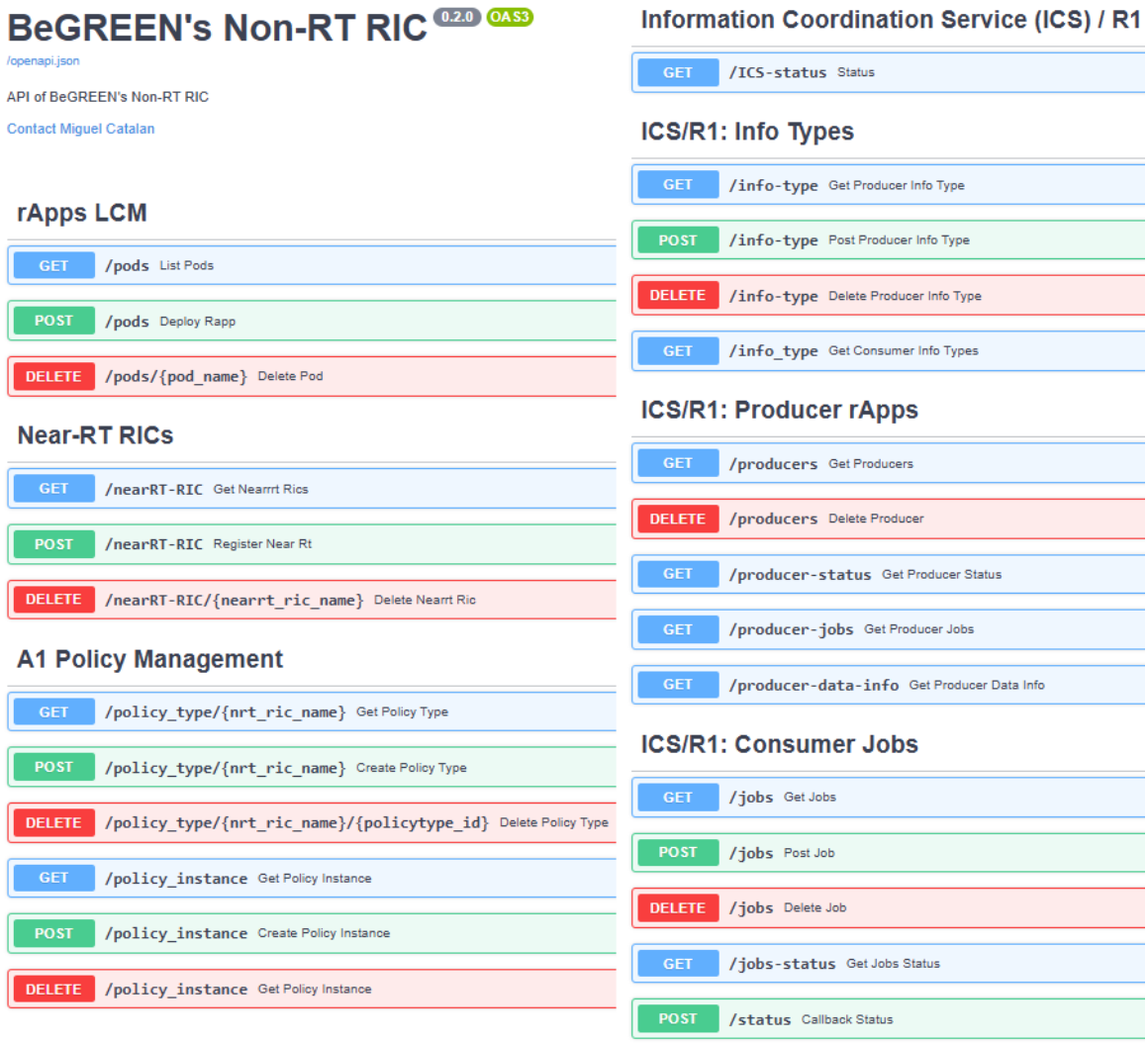


Figure 2-32: Validation of the Intelligence Plane - BeGREEN SMO and Non-RT RIC REST API - initial validation [1]

The first function allows to register the Near-RT RIC and specify the communication protocols. In the case of dRAX, Kafka is used and therefore, as depicted in Figure 2-33, the required topics to subscribe to KPM statistics and to send A1 policies are defined.

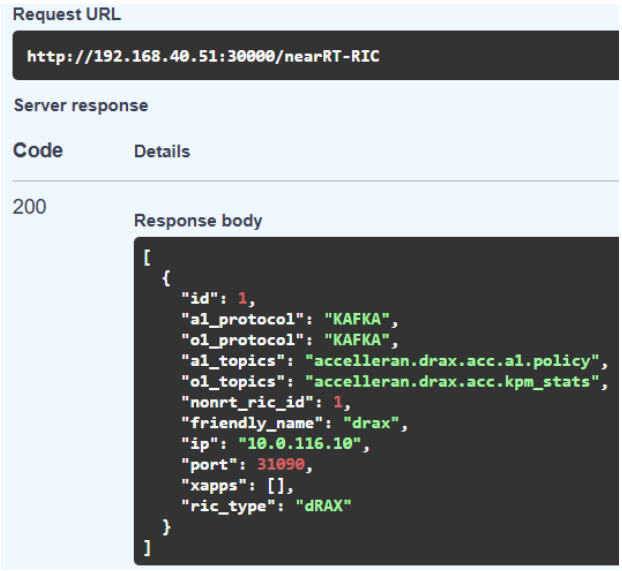


Figure 2-33: Validation of the Intelligence Plane - Near-RT RIC registration

Request URL

```
http://192.168.40.51:30000/policy_instance
```

Server response

Code	Details
202	Response body


```
{
  "policy_data": {
    "scope": {
      "cellIdList": [
        {
          "plmnId": {
            "mcc": "001",
            "mnc": "01"
          },
          "cId": {
            "ncI": 4886716418
          }
        }
      ]
    },
    "esObjectives": {
      "esPercentage": 80
    },
    "policyPrio": 100
  },
  "policytype_id": 1,
  "ric_id": "drax",
  "timestamp": "1741625801",
  "id": 1,
  "policy_id": "es_begreen",
  "policy_type_id": 1,
  "nearrt_ric_id": 1
}
```


Figure 2-34: Validation of the Intelligence Plane – Energy Saving A1 policy instance

The A1 policy management functions deal with the registration of A1 policy types and with the generation of A1 policy instances. While the registration of A1 policy types is usually done manually, A1 policy instances are generated and managed by the control rApps. In BeGREEN we focused on the implementation of the Energy Saving policy defined in BeGREEN D4.2 [2]. The policy extends the energy saving policy of O-RAN by introducing the Energy Score objective and the Policy Priority field. The schema of the policy, which is used to validate outgoing (Non-RT RIC) and incoming policy instances (Near-RT RIC), can be found in the Annex I. Figure 2-34 shows an example of an Energy Saving policy instance sent to a specific cell.

Regarding the collection of KPMs, the KPM producer rApp gathers this data from the dRAX Kafka Bus, exposing available metrics to any rApp requiring them. Figure 2-35 shows an example of the monitored cells and metrics during experiments.

Finally, we have also implemented the Energy Rating Assist rApp, being used to expose the energy rating function hosted in the AI Engine to other rApps. Figure 2-36 depicts the data model of the energy rating information type, which expects as inputs the list of cells, the period between ratings and the historical limit of period to be considered to compute the rating among cells. The output is generated by the energy rating function, as was defined in 2.2.1.

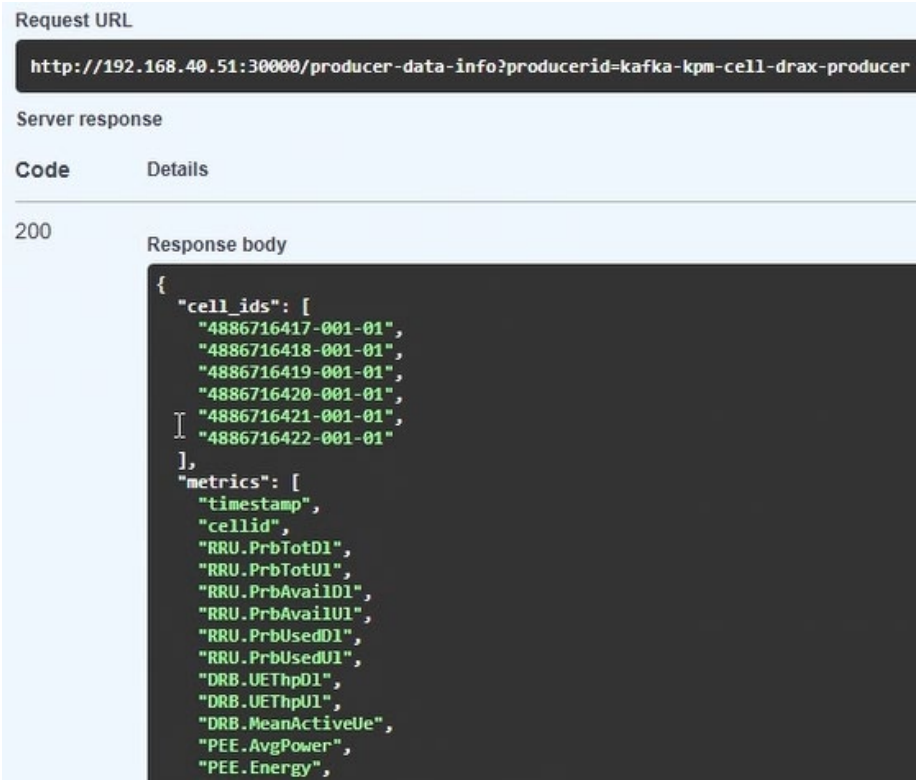


Figure 2-35: Validation of the Intelligence Plane – KPM producer rApp

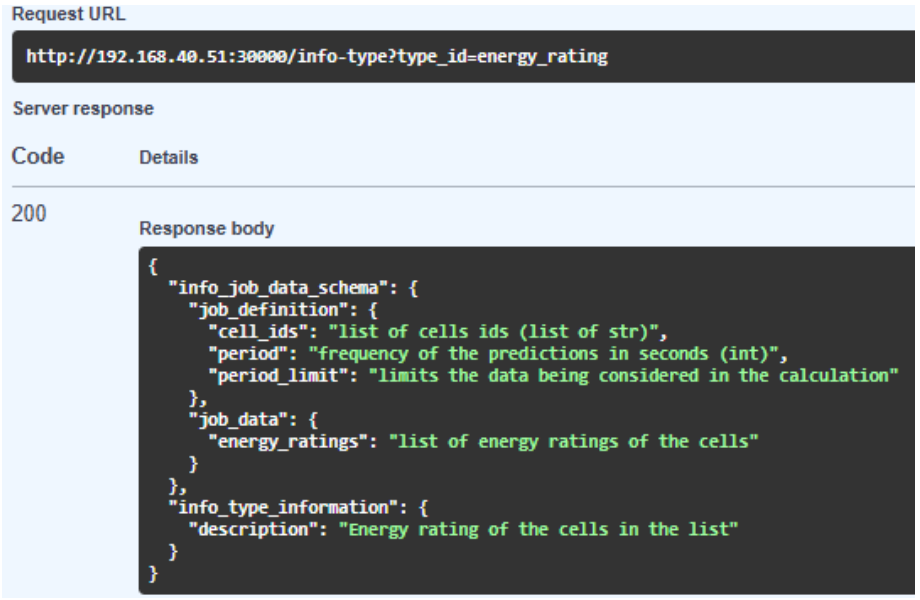


Figure 2-36: Validation of the Intelligence Plane – Energy Rating Assist rApp

2.3.5 Conflict mitigation

This section describes the validation of conflict mitigation within the Intelligence Plane scope. Figure 2-37 illustrates the dRAX conflict mitigation dashboard, which integrates the A1 policies received from the Non-Real-Time RIC. In the upper section, the dashboard displays the A1 policies applied to the system, detailing the scope of affected cells, objectives, and policy priorities, as outlined in BeGREEN D4.2 [2] and shown in Figure 2-38.

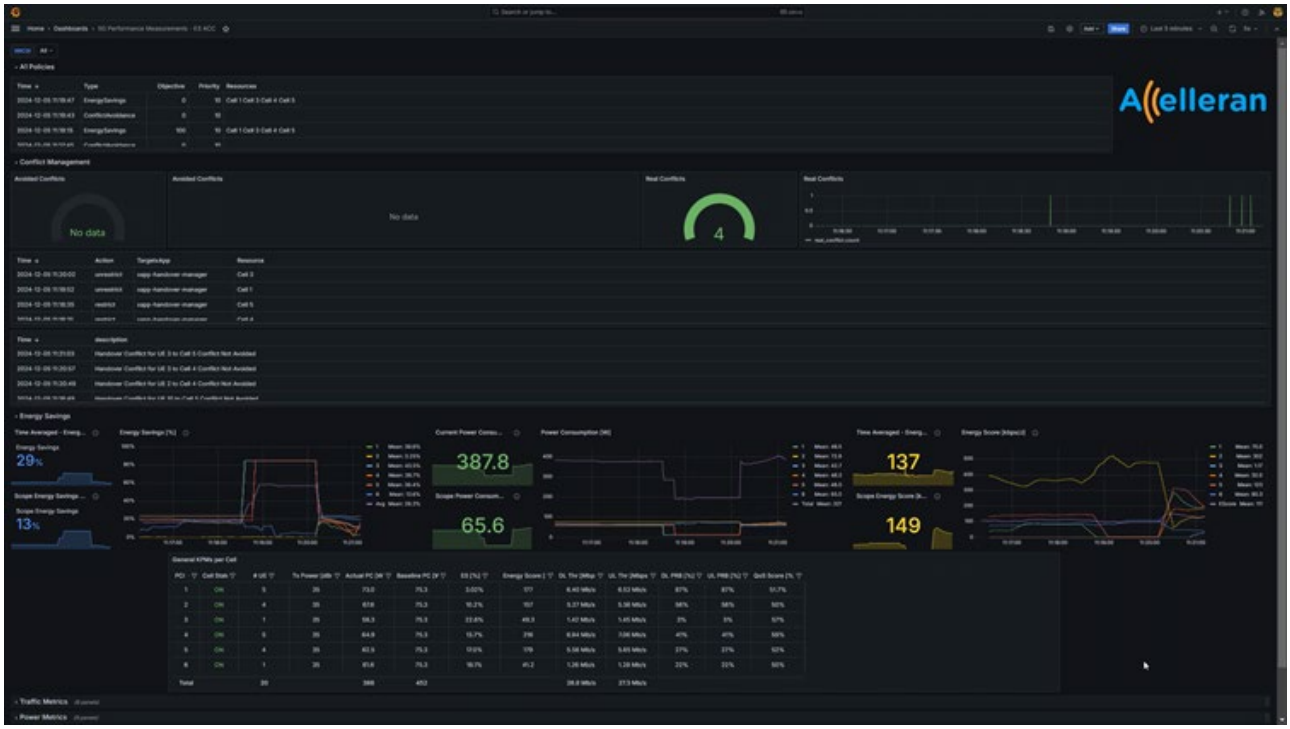


Figure 2-37: Conflict mitigation – Near-RT RIC Grafana dashboard - General Dashboard

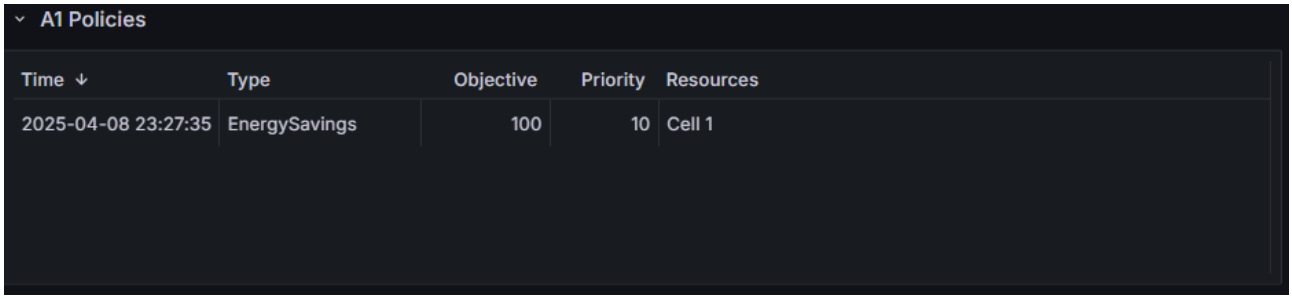


Figure 2-38: Conflict mitigation – Near-RT RIC Grafana dashboard - A1 policies section

The next section of the dashboard presents a comparison between the number of actual conflicts and the avoided conflicts, along with the timestamps of their occurrence. This is detailed in Figure 2-39. Further below, the conflict guidance messages are detailed, indicating whether a conflict was detected and if it was successfully mitigated. The bottom section of the dashboard displays network performance metrics, as described in BeGREEN D4.2 [2]. These include general KPIs per cell, such as energy savings, power consumption or the energy score, as depicted in Figure 2-40.

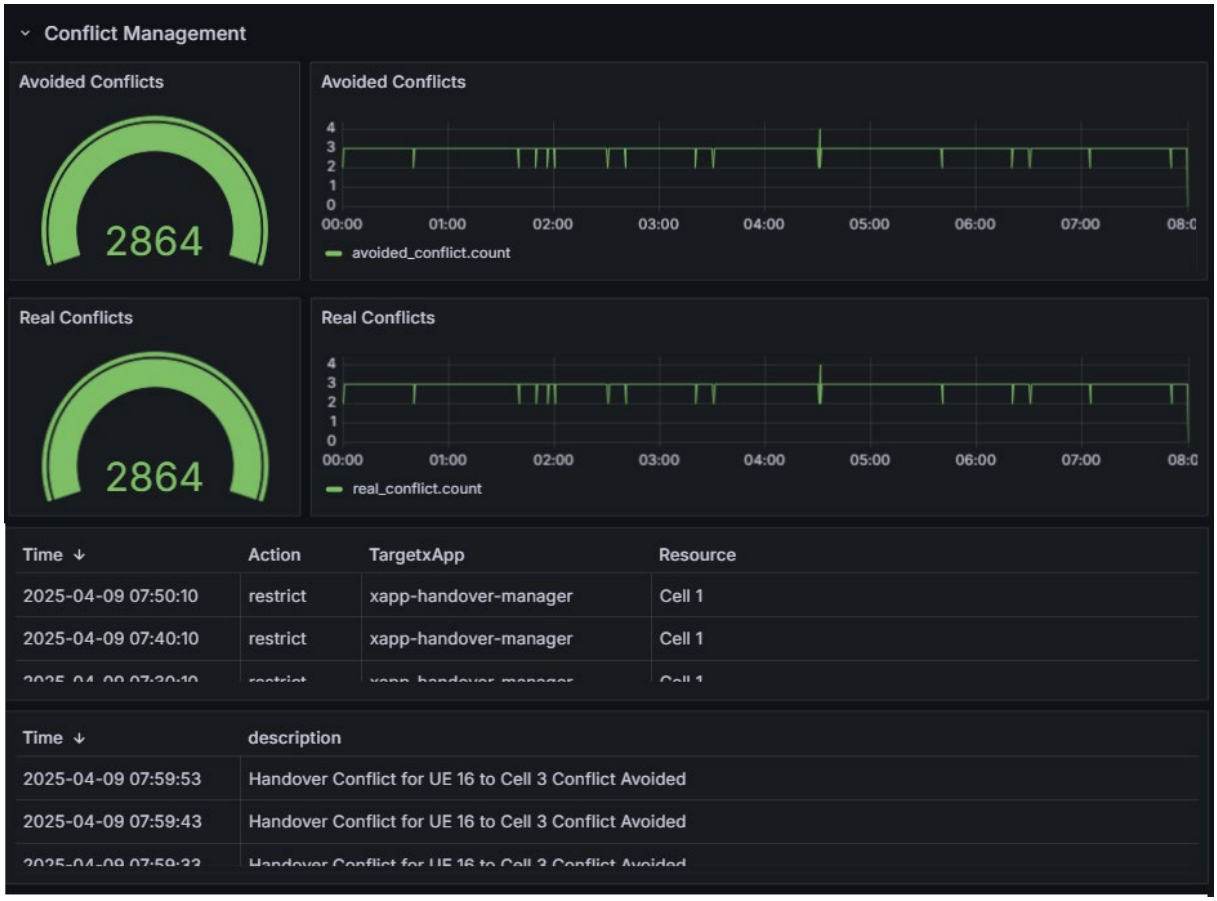


Figure 2-39: Conflict mitigation – Near-RT RIC Grafana dashboard - Conflict Management Section

Figure 2-40: Conflict mitigation – Near-RT RIC Grafana dashboard - General KPI per cell section.

For validation, we used the scenario described in BeGREEN 2.2 [20] and depicted in Figure 2-41, where six cells serve 20 UEs. The Non-RT RIC applies an energy efficiency policy of 100% to four cells (Cell 1, Cell 3, Cell 4, and Cell 5), reverting to 0% after two minutes. This transition is intentionally designed to trigger conflicts between the Energy Savings xApp and the Smart Handover xApp. To assess the effectiveness of the Conflict Manager, a set of measurements was conducted without the system activated, followed by another set with the Conflict Manager enabled, allowing for a direct comparison of its impact on conflict resolution.

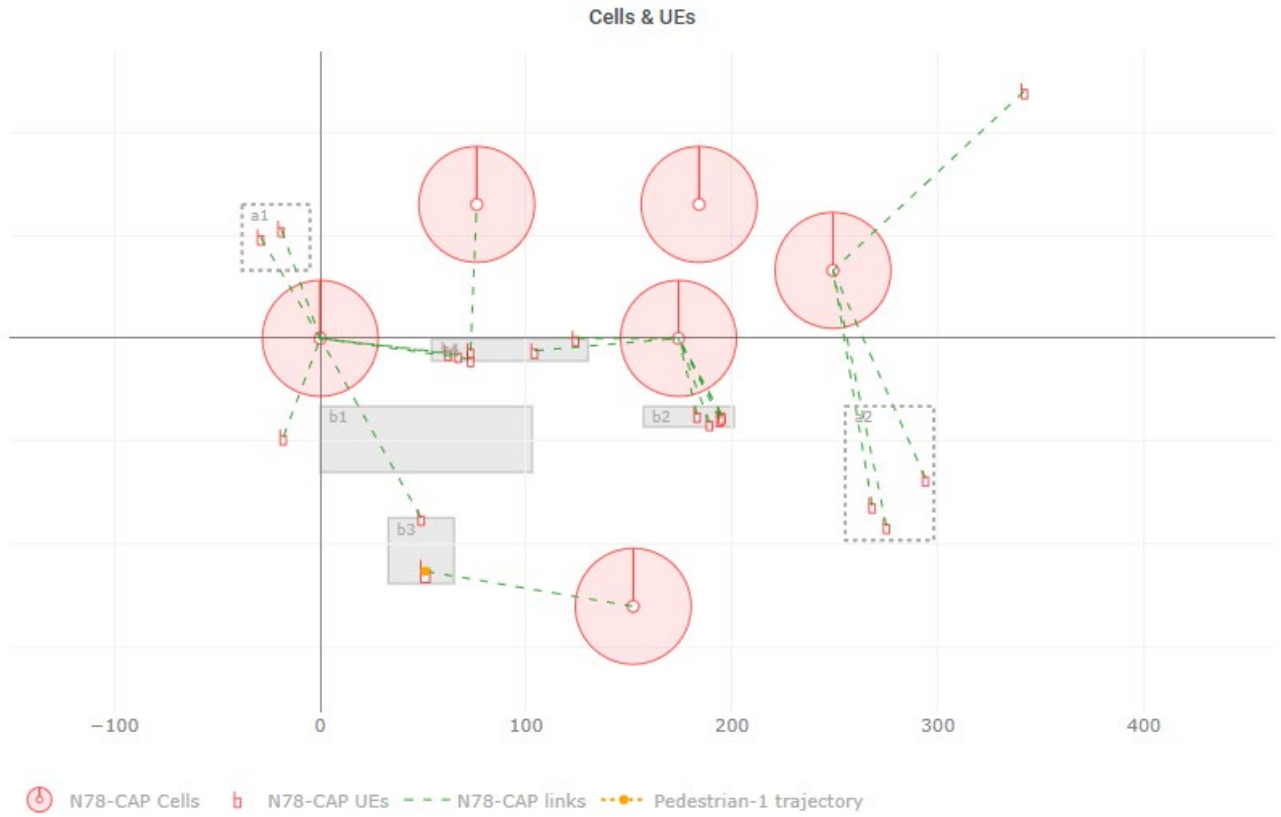


Figure 2-41: Conflict mitigation – Emulated scenario based on Adastral Park deployment

The Energy Saving xApp, upon receiving a policy of 100% of energy savings, emits a restriction message to the Smart Handover (SH) xApp to indicate that these resources (cells) are restricted. Similarly, when a policy of 0% energy savings arrives, the Energy Saving xApp issues an unrestricted message to the affected cells every time that the cell is being turned on. The restricted/unrestricted messages are within the authority of the xApps, following the internal policies of the Policy Handler. The target xApp must process these messages and implement a logic to handle the restriction accordingly. In this case, the SH xApp, after deciding which users should be handed over, double-checks the restricted cells. If the Conflict Management system is not in place, it proceeds as usual and issues the handover command. However, if the Conflict Manager is active, the command is not issued to the RAN. Additionally, if a resource is restricted, the SH xApp sends a conflict detected message to the conflict detection system. When the Conflict Manager is in place, this message is labelled as “Conflict Avoided”; otherwise, it is marked as “Conflict Not Avoided”, as shown in Figure 2-42.

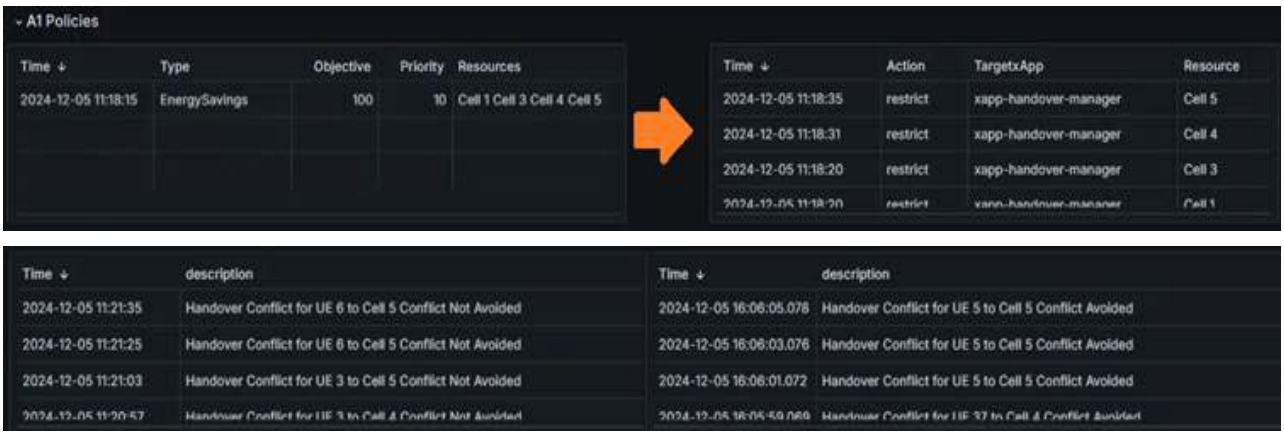


Figure 2-42: Conflict Mitigation - Example of conflict detection and avoidance

Figure 2-43: Conflict Mitigation – Example of application and impact on energy efficiency

2.3.6 RIS validation

As discussed in section 2.1.4, besides the opportunities brought by smart surfaces in communication systems, RIS have also emerged as a promising ISAC building block for next-generation indoor positioning systems. Awareness of UE location and mobility, allows for a considerably more effective deployment of these RAN energy-saving strategies, culminating in reduced transmission overhead and optimized power allocation. For example, ISAC positioning solutions can enable a 50% reduction in beam measurements for energy-efficient beam management and facilitate decreased Channel State Information (CSI) reporting, which conserves power system-wide.

Current indoor positioning solutions using smart surfaces estimate user locations by dynamically probing different reflection configurations from a precomputed/pre-calibrated codebook, searching for configurations that maximize the gain of the received communication signal [21][22]. However, current approaches face several critical challenges that hinder their performance and scalability such as i) reduced resolution because of the space constraints and sub-6GHz wavelength, ii) low-gain changes due to the predominance of the direct path, and iii) phase misalignment because of the interferences that may be created at the receiver between the reflected signal and the direct path.

To showcase the integration of RIS with the BeGREEN Intelligence Plane Architecture, we present here an RIS-enabled ISAC solution called “Echoes” that addresses the previous challenges. Echoes is an indoor location system that leverages the reconfiguration capabilities of smart surfaces as well as the flexibility of future mobile systems based on O-RAN architecture, such as the BeGREEN architecture. In this sense, Echoes is able to provide the position of one (or more) UEs connected to a gNB (in our case, a 5G small cell) with the help of a RIS in an indoors scenario in the sub-6GHz band, where the RIS gain is low, and the direct path is strong.

To do so, Echoes infers the received signal phase solely from power measurements of reference signals used to measure the uplink channel (more precisely using the metric L1M-UL-SRS-RSRP available from the E2SM-KPM). Echoes achieves this by exploiting an inherent property of RIS: steering configurations are not singular, and introducing a constant phase offset to all elements achieves the same intended reflection direction but

with a different reflected phase. In this line, Echoes combines the sensed direction of a user with the angle-of-arrival (AoA) estimation to estimate the position of a user in a sensing area.

2.3.6.1 System Model

Echoes employs a model-based approach for indoor localization, leveraging the combined capabilities of a small cell and a RIS to estimate the position of users within a sensing area. As depicted in Figure 2-44, we consider a user (TX) within the sensing area along the direction (θ_u^R,φ_u^R) with respect to the RIS reference. Also, the same user is oriented horizontally with an angle, Θ_u^h, and vertically with an angle, Θ_u^v, from the small cell receiving antenna array.

To know how to configure the smart surface, we can derive the optimal phase shifts that the unit cells of a smart survey should apply to reflect an incoming signal from (θ_t,φ_t)towards (θ_r,φ_r) by compensating their phase difference. This can be done by modelling the smart surface configuration with its corresponding steering vector model, which can be summarized as:

$$\theta_n = \arccos(cos\theta_t - cos\theta_r) \quad (2\text{-}2)$$

$$\varphi_n = \arcsin(sin\varphi_t - sin\varphi_r) \quad (2\text{-}3)$$

To navigate the possible combinations of unit cells' phase shift configurations, it is a common practice to compute a configuration codebook, which is a set C = {C_k}, where |C| = K. This configuration codebook consists of precomputed phase shifts configurations that allow steering an incident wave (θ_t,φ_t) towards a specific direction (θ_r,φ_r). Codebook configurations C_k are generated by discretizing a range of values for θ_n and φ_n. Please note that the unit cell configurations are not singular, so if we apply the same phase offset ψ_0 to all cells, we still maximize the power of the reflected signal, but we are changing its phase.

Echoes periodically probes all the configurations C_k of the smart surface's codebook using a finite set of Ψ offset values for each configuration. Since, in mobile networks, users periodically transmit reference signals in the uplink, the network constantly measures the received power per receiving antenna to estimate the quality of the channel. In 5G-NR, UEs periodically transmit the Uplink Sounding Reference Signal (UL SRS) on the uplink, allowing the serving gNB to estimate the quality of the channel at different frequencies. UL SRS are periodically transmitted as often as every 2nd subframe (2 ms) or as infrequent as every 16th frame (160 ms). The UL SRS are transmitted on the last symbol of the subframe.

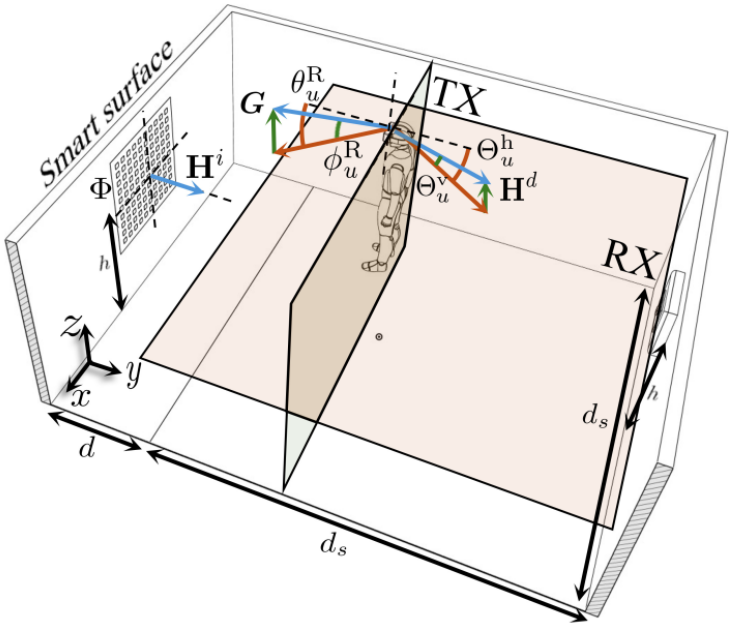


Figure 2-44: RIS Validation - Echoes Scenario

Based on the received power measurements from the UL SRS, the serving gNB schedules uplink transmissions on resource blocks of good quality. O-RAN supports monitoring the values of the received power of the UL SRS [23][24], measuring them periodically for the different UEs and antennas in the serving gNB.

Echoes uses these measurements to estimate the position of a user in three steps:

- Step 1: The first step is required to calculate the direction of the receiver from the RIS point of reference $(\theta_u^R, \varphi_u^R)$ To do so, Echoes searches for the set of configurations C* = $\{C_k(\psi_0^m)\}$ that maximizes the RSRP in each antenna m of the small cell. Afterwards, the M measurements using (2-2) and (2-3) are averaged to estimate $(\theta_u^R, \varphi_u^R)$. We assume the angle RIS – gNB $(\theta_s^R, \varphi_s^R)$ is known.
- Step 2: The second step relies in the concept that the difference between the estimated offset values that maximize the RSRP in each antenna pair m and m' (with the best C_k), is approximately the phase difference of a signal arriving to m and m'. Echoes exploits this observation to predict the AoA between the small cell and the user, and obtain Θ_u^h and Θ_u^v using:

$$\Theta = \arcsin\left(\frac{\angle s_m - \angle s_{m'}}{\pi}\right) \quad (2\text{-}4)$$

  (2-4) is based on the fact that a wave traveling an extra distance d changes its phase by an amount $\Delta\psi = 2\pi\frac{\lambda}{d}$, which is equal to the phase difference between the signal s arriving at antenna m, and the signal s arriving at the antenna m', i.e., $\Delta\varphi = \angle s_m - \angle s_{m'}$. Echoes calculate this phase differences by adjusting the offset $\psi_0$of the best C_k to maximize a fitted sinusoidal function. The offset ψ_0 that maximizes the RSRP value in that function, will align the phase of the reflected wave with the phase of the direct wave, i.e.:

$$\psi_0^m - \psi_0^{m'} = \psi_d(u, s_m) - \psi_d(u, s_{m'}) + \psi(s_m) - \psi(s_{m'}), \quad (2\text{-}5)$$

  where $\psi_d(u, s_m) - \psi_d(u, s_{m'})$ is the phase difference of the signal s arriving at antenna m and m' respectively – which we can use directly to compute (2-4) – and $\psi(s_m) - \psi(s_{m'})$ is the difference between the phase alteration induced by the multi-path effect. Here we assume that the direct path between the TX and the RX dominates any multi-path contribution and that to reduce the effect of the multi path, we compute multiple estimations of $\psi_0^m - \psi_0^{m'}$.
- Step 3: Once these angles have been calculated (Θ_u^h, Θ_u^v) from Step 2 and $(\theta_s^R, \varphi_s^R)$ from Step 1, Echoes geometrically locates the user within the sensing area, searching the best position (x,y,z) that minimizes the sum of the mean squared error $|e|^2$ of all the angle estimations. To do so, Echoes follows an optimization problem where all solutions lie in the Euclidean space. Since $|e|^2$ is continuous and convex in the sensing area, it can be solved using numerical methods.

2.3.6.2 O-RAN Integration

Echoes considers an O-RAN enabled indoor scenario that consists of a gNB (small cell) covering the desired sensing area and a RIS that allows for the estimation of the users' positions. To integrate the RIS into the BeGREEN architecture, we follow the approach described in BeGREEN D4.2 [2] and D3.3 [10], as illustrated in Figure 2-45.

First, we consider an external application that requires user location to be deployed in SMO (e.g., as an rApp) or outside of the RAN architecture. That positioning application interacts with the Echoes xApp, which is deployed in the near-RT RIC and implements the core functionalities of Echoes as described above The Echoes xApp performs the following functions: i) receiving L1M-UL-SRS-RSRP measurements from each small cell antenna via the E2 interface, ii) reconfiguring the smart surface with a given configuration, and iii) estimating user positions once Echoes has collected all the necessary measurements.

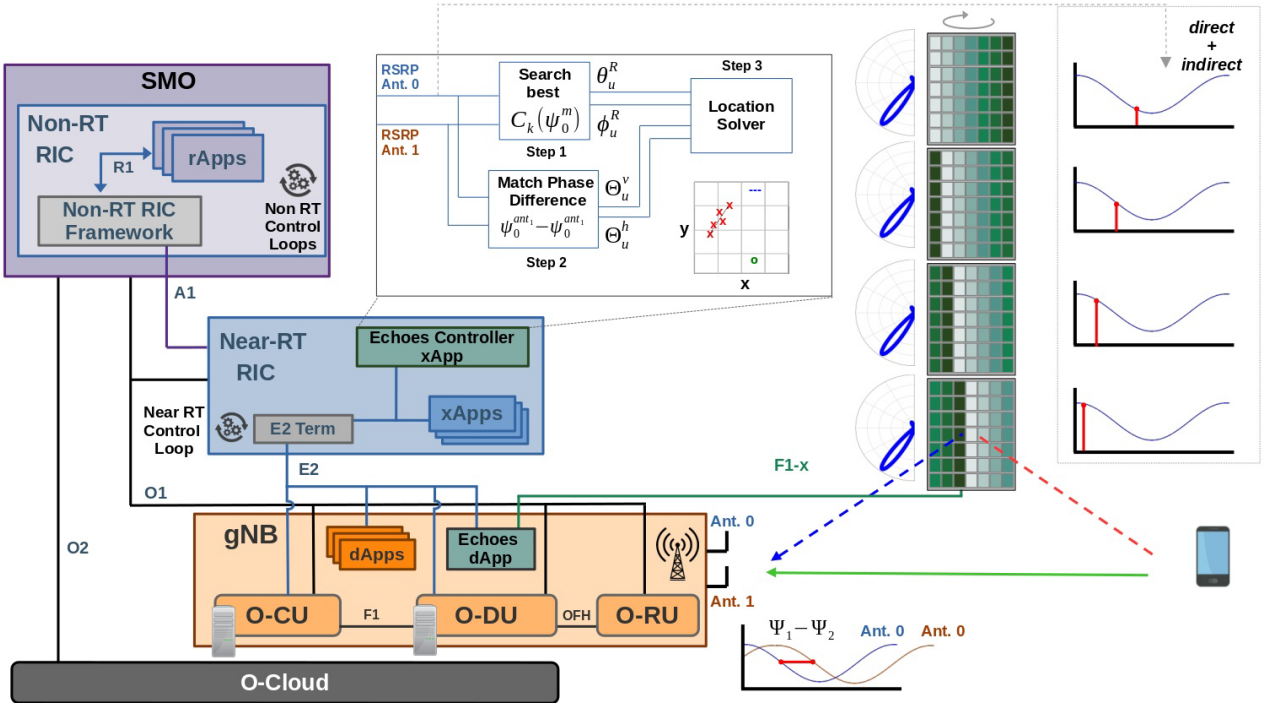


Figure 2-45: RIS validation - Echoes' O-RAN architecture integration and intuitive description

For the use case of Echoes there is not direct RIS control from the non-RT RIC or the near-RT RIC, meaning that although defined by the BeGREEN architecture, O1+ and E2+ are not used. Instead, the control is performed in-site through the gNB (small cell). The small cell, which may be deployed in edge O-Cloud platform is connected to the RIS through the F1-x interface [25] (a wired connection, such as an Ethernet or USB cable). Within the small cell, there is deployed a dApp [26] that enables the reconfiguration of the smart surface locally according to a predefined codebook configuration and offset $C_k(\psi_0)$.

To describe the Echoes positioning procedure, Figure 2-46 depicts Echoes operational workflow, which consists of the following four phases: 1) *calling the Echoes* in which the xApp receives a location subscription request, 2) *listening to the Echoes* in which the xApp subscribes to the power measurements of the uplink reference signal periodically sent from a user of the small cell, 3) *tuning the Echoes* in which the xApp reconfigures the smart surface via the dApp, and 4) *grasping the Echoes* in which the xApp runs the optimization algorithm to locate a user. In the following we detail each step:

- *Calling the Echoes:* Echoes' workflow starts when the xApp receives a location subscription request from the external application or rApp whose objective is to locate a user within the sensing area. The application may send a positioning service request using the A1 interface.
- *Listening to the Echoes:* Echoes's xApp subscribes to the periodic power level measurements of UL SRS per receiving antenna measured from the target UE. This measurement is available in the E2 interface through the E2SM-KPM [27]. Therefore, Echoes uses the E2 interface to request the small cell the periodic metric reports of this metric with a Report Subscription Request, which the small surface will provide in the form of Report Indications.
- *Tuning the Echoes:* Echoes' core location estimation procedure is based on iteratively reconfiguring a smart surface according to an algorithm implementing Steps 1 and 2 as described above, while compiling all the necessary power measurements to infer the position of a user within the sensing area. To reconfigure the smart surface, Echoes' xApp uses the proxy-like Echoes dApp co-located with the small cell. The dApp locally controls the smart surface at site, reconfiguring all its elements through the F1-x interface [28]. Echoes sends to the dApp the index k of a codebook configuration

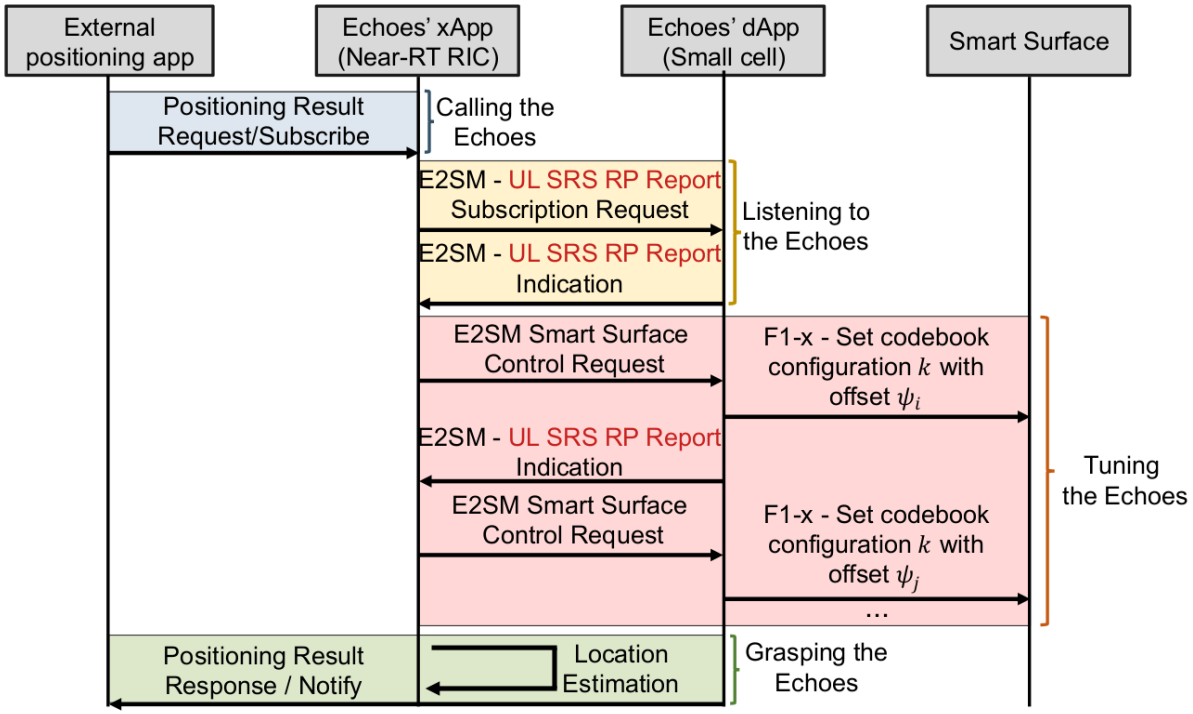


Figure 2-46: RIS validation - Echoes workflow

$C_k(\psi_0)$ and an offset value ψ_0 to be applied. The interface between Echoes's xApp and the dApp uses the standard E2 interface leveraging the E2AP sets of predefined procedures and services that enable the control communication. Over E2AP, Echoes expands E2SM enabling two smart surface-specific service models, E2SM-SSC and E2SM-SSM, which were introduced in BeGREEN D4.2 [2] and D3.3 [10]. On one hand, E2SM-SSC allows for setting a given codebook configuration in the smart surface specifying azimuth, elevation and offset, i.e., $C_k(\psi_0)$. On the other, E2SM-SSM supports the reporting of the status of the RIS, all supported codebook configurations and the current codebook setting currently configured.

- *Grasping the Echoes:* Once Echoes has received all the necessary power level measurements, Echoes estimates the position of a user according to above description. Echoes procedure runs periodically to progressively improve the sensed positions, periodically sending the obtained position estimations to the subscribed application.

2.3.6.3 Echoes Results

To assess Echoes performance realistically, we have established an experimental testbed. Figure 2-47-a illustrates the indoor testbed schematically, while Figure 2-47b illustrates a real deployment.

The TX and RX were developed using two USRPs B210 devices operating with srsRAN [29]. The TX communication stack executes on a desktop PC, while the RX's Distributed Unit (DU) and Central Unit (CU) run on the O-cloud edge server. Both RF front-ends operate at 5.3 GHz (wavelength λ = 5.65 cm). The RX uses two omnidirectional low-gain antennas spaced λ/2 apart, whereas the TX employs directional horn antennas. These horns were chosen to minimize scatterers in the testbed and ensure experimental consistency and reproducibility. The O-cloud server also hosts the O-RAN SC Near-RT RIC (i-release) [30], where Echoes operates as an xApp. Echoes gather uplink power measurements per antenna from the RX. The server connects to the smart surface via a USB interface (acting as the F1-x interface) and runs a dApp to manage codebooks and surface configurations.

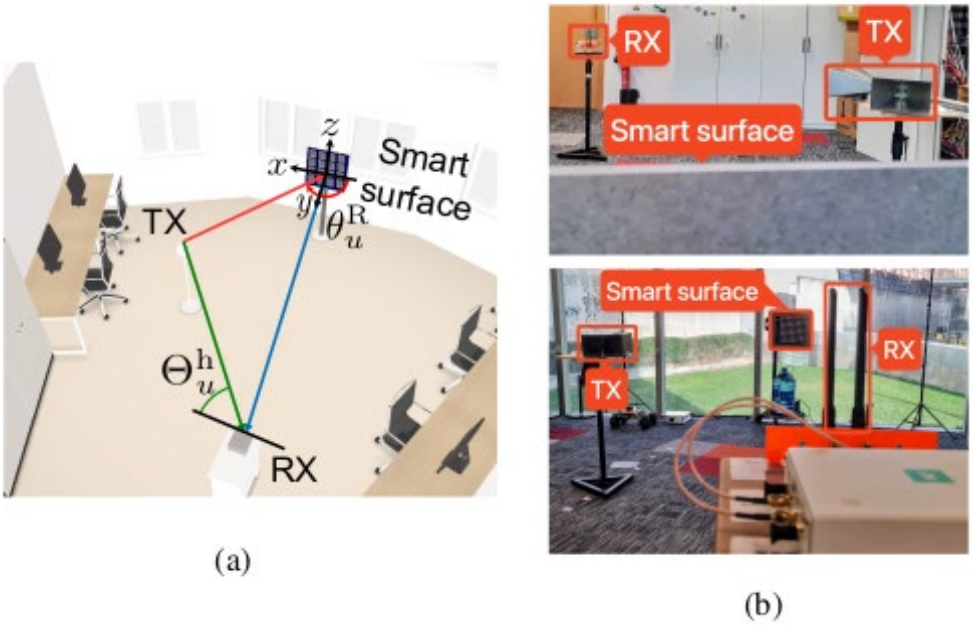


Figure 2-47: RIS validation - Illustrations for the, a) testbed scheme design, and b) real deployment

The smart surface is strategically positioned near the front wall to maximize spatial coverage for user localization. The RX (small cell) is placed at the opposite end of the room, aligned centrally with the surface to ensure broad coverage. The TX is situated between the surface and RX, with flexibility to reposition it for varied measurements. Notably, the TX, RX, and smart surface lie in the same horizontal plane, enabling position estimation using the two-antenna array. Notice that we only probe configurations for the different values of the azimuth θ as having the TX and RX in the same plane sets the optimal elevation to ϕ = 0º. Echoes estimates the position using (θ_u^R , Θ_u^R).

The smart surface prototype in our testbed features a 10×10 grid of unit cells based on patch antennas, operating at 5.3 GHz. The patch antennas are spaced δ = λ/2 = 2.82 cm apart. Each unit cell supports a 3-bit phase shift configuration: the **000** setting deactivates the element, while the remaining 7 bit combinations apply phase shifts $\psi_n = (2\pi/7)_n n$ for n ∈ {0, ..., 6}, corresponding to ψ_n ∈ {0°, 51.42°, 102.85°, 154.85°, 205.71°, 257.14°, 308.57°}. Deactivating all elements switches the surface to absorption mode. The prototype achieves a half-power beamwidth (HPBW) of ≈10° when reflecting signals toward a target direction.

A far-field precomputed codebook manages configurations. Since each element only supports 7 discrete phase shifts, codebook values are rounded to the nearest ψ_n. When applying a global phase offset, it is constrained to multiples of ψ_n (n ∈ {0, ..., 6}). The codebook spans azimuth angles [−60°, 60°] and elevation angles [−45°, 45°], incremented in 5° steps, yielding 437 total configurations. Figure 2-48a demonstrates a configuration with θ = 30°, ϕ = 0°, and no offset, while Figure 2-48b -illustrates the same beam direction with an added ψ_o = 51.42° offset.

The first tests consist of evaluating the effect of adding a phase shift offset to all elements. The effect of adding an offset value creates a sinusoidal pattern on the received power gain. Figure 2-49 shows the normalized UL-RSRP measured for the two antennas at the RX. It is possible to clearly observe the sinusoidal pattern that different offsets create on the received gain. We also see that both sine functions are not in-phase and precisely that observation is what allow us to measure the angle of arrival between a user and the receiver.

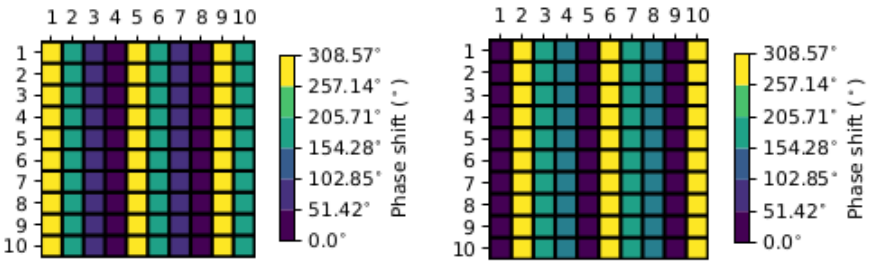


Figure 2-48: RIS validation - Codebook configuration with, (left) θ = 30º, ϕ = 0º and ψ = 0º, and (right) with θ = 30º, ϕ = 0º and ψ = 51.42º

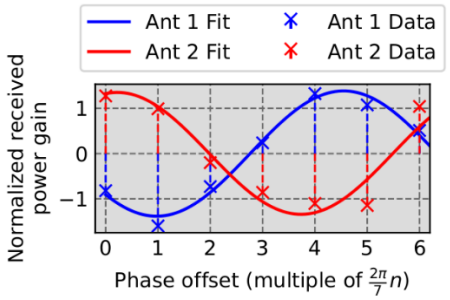


Figure 2-49: RIS Validation - Received power gain per antenna using the best phase shift offset value

To test the localization performance of Echoes, we place the TX at multiple locations and analyze the UE positions estimated by Echoes. Figure 2-50 marks the actual UE positions with black dots, covering angular ranges $\theta_u^R \in [215°, 245°]$ and $\Theta_u^h \in [27°, 73°]$, with distances from 1.3 m to 4.3 m relative to the smart surface. Since all components lie in the same plane, we limit Echoes' codebook probing phase to configurations C_k with $\theta \in [-55°, 55°]$ and $\phi = 0°$. Figure 2-50 also shows Echoes' predicted locations (red crosses) for each TX position. The estimates align closely with the TX-to-smart-surface axis, indicating strong θ_u^R accuracy. However, minor variability arises along the TX-to-small-cell axis due to less precise Θ_{hu} calculations. Most predictions exhibit a systematic southward bias, with two outliers deviating from this trend.

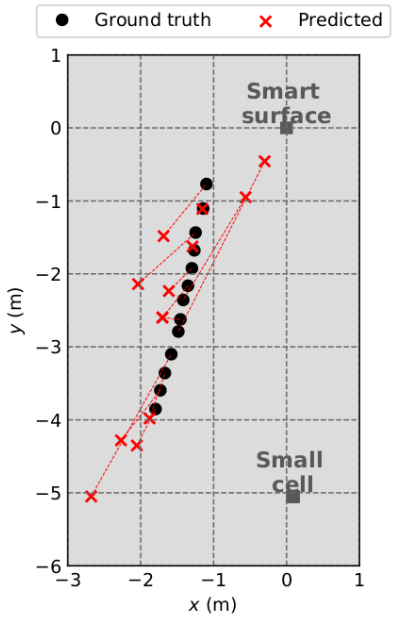


Figure 2-50: RIS validation - Predicted UE positions by Echoes in the 2D space.

The estimation error is due to two main factors. First, the codebook's steering angle discretization of 5º ultimately limits the accuracy of estimating θ_u^R, resulting in a Mean Absolute Error (MAE) for θ_u^R is 2.15º. Second, the fact that we are only using one pair of antennas makes that the Θ_u^h error is relatively high (with a MAE of 14.15º), limiting our ability to locate the TX in the different tested position. Using additional pairs of antennas would help us to reduce this error. Nevertheless, these errors allow Echoes to give a location estimation with a MAE distance error of 0.92 m.

2.3.6.4 Conclusion

To validate the integration of RIS into the BeGREEN architecture, we have presented Echoes, an ISAC indoor localization system, which leverages the available metrics of an O-RAN small cell and a RIS to predict user's positions within a sensing area. Precise user location is pivotal for enhancing 5G RAN energy efficiency, as it may enable critical power-saving mechanisms like adaptive beamforming, which focuses energy towards specific UEs, and intelligent cell sleeping based on real-time UE distribution.

Echoes relies on the UL-RSRP signals and the available periodic KPM measurements in O-RAN to locate a user by suitably reconfiguring the smart surface using both the E2 and F1-x interfaces. Echoes adjusts the smart surface's phase offsets to identify the user direction that optimizes receiver gain and compute the AoA between the user and small cell. We evaluated Echoes in a practical indoor 5G setup compatible with the BeGREEN architecture, highlighting its performance in uncontrolled, multipath-rich environments. Finally, we have detailed Echoes' integration into the O-RAN framework, demonstrating its operational viability under real-world architectural constraints. The use of these UE location and mobility awareness solutions such as Echoes, within the BeGREEN architecture, may allow for significantly more effective implementation of RAN energy-saving strategies and lead to reduced transmission overhead or optimized power allocation.

3 Final Evaluation of AI/ML-Assisted Procedures to Enhance Energy Efficiency

This chapter presents the final evaluation of the AI/ML-assisted methods developed throughout this WP. Compared to previous deliverables, and for having a more consistent structure in this final deliverable, RIS and Intelligence Plane validations have been reported in Chapter 2. Additionally, contributions related to energy efficiency in ML models and the Intelligence Plane are detailed in Section 2.2. Consequently, this chapter focuses on the final validation of five AI/ML-assisted methods aimed at enhancing energy efficiency across the BeGREEN use cases: vRANs, real 5G NSA deployments, relay-enhanced RANs, edge nodes hosting UPFs, and edge nodes hosting AI services. Results highlight the benefits of the developed solutions and the application of AI/ML to enhance energy efficiency without impairing traffic performance.

3.1 Compute resource allocation in vRAN

As we explained in previous deliverables, the increased computing overhead has its roots in the lack of cache memory isolation. The computing overhead increases the total energy consumption of the vRAN platform. Figure 3-1 shows the relationship between the normalized energy consumption on top of the system's baseline (i.e. idle) consumption of our vRAN platform as a function of the total computing load. The computing load and the energy are linearly related. Therefore, it is key to minimize the computing usage of our vRAN platform to keep operational energy costs low. In the previous deliverables, we developed a solution to dimension the computing capacity of a vRAN system, considering the increased computing usage due to the noisy neighbour problem.

However, in previous deliverables we did not consider minimising the increased overhead but adapting to it. In this final deliverable, we seek to reduce the computing overhead as much as possible. We begin studying how isolating the different cache memory levels influences the total system computing consumption. We found that vBSs traffic demand is virtually orthogonal to using cache resources, i.e., vBSs use as much cache memory as possible. However, the utility of cache memory is different for different traffic demands. The vBSs with higher demands and Signal-to-Noise-Ratio (SNR) can reduce their computing usage more when they have more cache memory available. Thus, developing a solution that strategically allocates the cache memory resources to the different vBS instances according to their demands is key to minimize energy consumption. Due to the complex relationship between computing and radio resources, we propose a novel approach that can effectively allocate the cache resources to minimize the total computing usage and, consequently, reduce the energy consumption of vRAN platforms.

3.1.1 Cache memory isolation

To evaluate the impact of cache memory resources on the noisy neighbor problem in vRANs' energy consumption, we measured the computing usage and low-level cache metrics deploying a different number

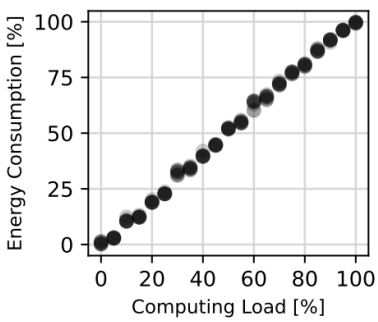


Figure 3-1: vRAN resource allocation - Energy consumption as a function of the computing load.

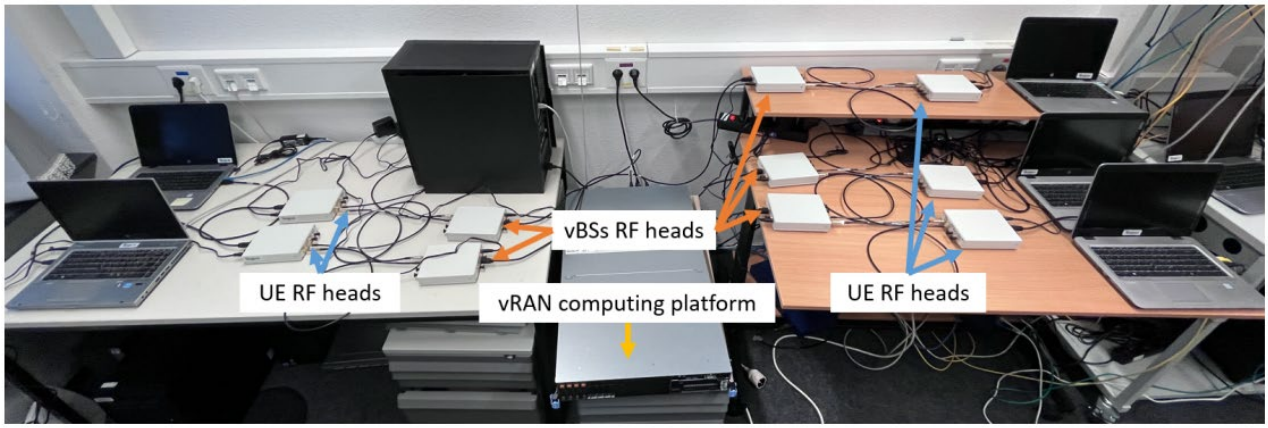


Figure 3-2: vRAN resource allocation - vRAN testbed

of vBS instances in different scenarios in our vRAN platform. Figure 3-2 shows our experimental vRAN platform. Therein, we deploy the different vBS instances in an isolated pool of computing cores from an Intel Xenon E5-2650 v4 CPU @ 2.20GHz in a shared off-the-shelf server.

Each vBS has its dedicated RF radio head connected to one UE, which is used to emulate the aggregated cell load. We configured the pool of computing cores to retain as much predictability as possible: (i) we isolated 12 cores with 12 dedicated Last Level Cache (LLC) ways, (ii) the system can only use C0/C1 C-states and we turned off hardware P-states and (iii) we deactivated Hyper-threading. We then initiate bidirectional data flows, both uplink (UL) and downlink (DL), with maximum load and good wireless channel conditions between each vBS instance and different user equipment (UE). We consider the following scenarios:

- **Ideal**. We compute the CPU usage scaling linearly the usage of a single vBS instance by assuming that the cache memory size also scales linearly.
- **No isolation**. We deploy an increasing quantity of vBS instances without any cache memory isolation mechanism.
- **Pinning**. L1 and L2 cache levels are dedicated per core. We pin different vBS instances to distinct cores to assess the impact on L1 and L2 cache isolation. To facilitate comparisons, the number of cores assigned to each vBS is given by:

$$\text{cores per vBS} = \left\lfloor \frac{\text{total cores}}{\text{deployed vBSs}} \right\rfloor$$

  where the total number of cores equals 12 and the number of deployed vBS increases from 1 to 5. Note that when 5 vBSs are deployed, each vBS is assigned to 2 cores and two free cores are left.

- **Pinning + LLC isolation**. We perform the L3 cache allocation in the same manner as the CPU pinning. We allocate the total number of cache ways equally for every single vBS, i.e. the number of cache ways per vBS is given by:

$$\text{cache ways per vBS} = \left\lfloor \frac{\text{total cache ways}}{\text{deployed vBSs}} \right\rfloor$$

  In our platform, there are a total of 12 cache ways. We use Intel CAT to allocate the LLC cache ways to the corresponding computing cores.

Figure 3-3 shows the measured computing usage in the different scenarios. We observe that, for the no isolation configuration, the computing usage increases by approximately 50% compared to the ideal case, increasing the energy consumption. This also impacts the IPC and MPKI, as shown in Figure 3-4and Figure 3-5, respectively. We observe that the IPC decreases and the cache misses increase as more instances are

deployed. There is a sixfold increase in the cache misses when transitioning from 1 to 5 vBSs. This increase also impacts the number of cycles required to execute the same number of instructions, decreasing the IPC.

The Pinning configuration shows a lower computing usage than the No isolation but still higher than the ideal case. The computing usage decrease is significant considering that L1 cache ways are approximately 100 times lower in size than L3 cache ways and L2 cache ways are 10 times lower in size than L3 cache ways. In Figure 3-4 we can observe a higher IPC across any number of deployed vBSs compared to the No isolation scenario. However, Figure 3-5 shows the same number of cache misses for all the cases. This might seem shocking at first glance, but as we have previously detailed L1 and L2 cache way size is very low compared to L3.

Finally, to study the effect of the L3 cache isolation. In Figure 3-3, the Pinning + LLC isolation configuration does not improve the computing usage. Specifically, the computing usage is the same for all the cases except for the case with 5 vBS, in which it is slightly higher. The reason behind this behavior is that we only allocate 10 out of 12 L3 cache ways to all vBSs for the case of 5 vBSs, while in the Pinning configuration, all vBSs have all the L3 cache memory available, using on average 2.4 L3 cache ways.

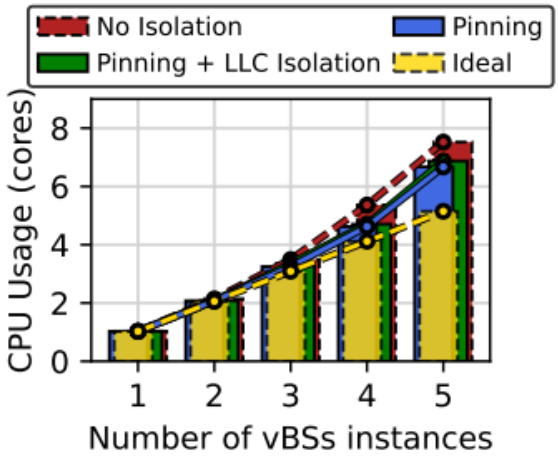


Figure 3-3: vRAN resource allocation - Comparison of the aggregated per-core usage with # of vBS instances showing the "No isolation", the "Pinning" and the "Pinning + LLC isolation" scenarios.

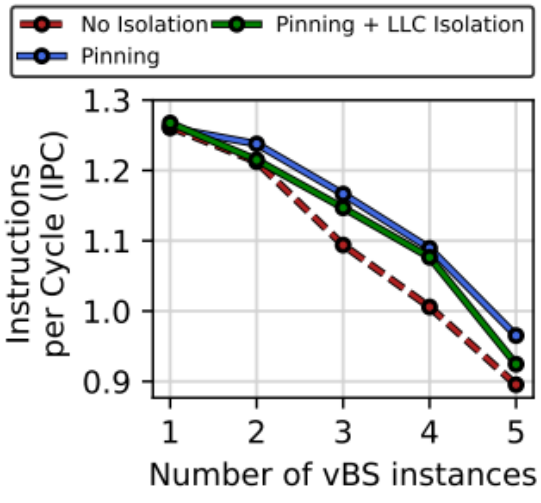


Figure 3-4: vRAN resource allocation - Instructions per cycle (IPC) with # of vBSs.

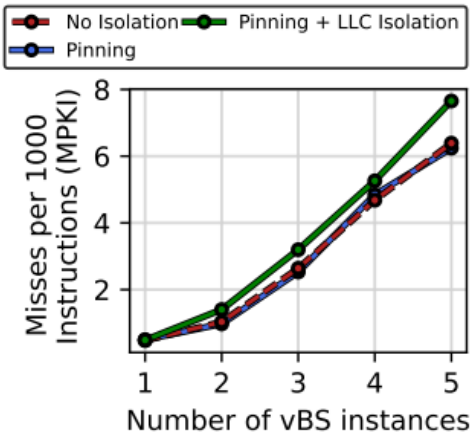


Figure 3-5: vRAN resource allocation - Misses per 1000 instruction (MPKI) with # of vBSs.

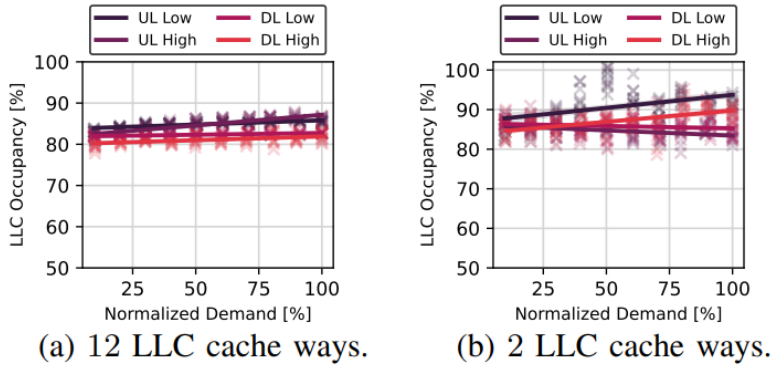


Figure 3-6: vRAN resource allocation - LLC occupancy in % as a function of the total demand for different SNR cases in UL and DL.

3.1.2 LLC Occupancy and utility

We now study how the allocation of LLC cache ways impacts the computing usage of a vBS instance. First, we measure the percentage of LLC cache memory used of the total LLC memory allocated to a vBS as a function of the traffic demand. Figure 3-6shows the LLC occupancy with 12 and 2 LLC cache ways for different SNR values and traffic demands in UL and DL. The high series depict the LLC occupancy with a high SNR environment while the low series depict the LLC occupancy with a low SNR environment. We can see that the total LLC occupancy is above 80% for uplink and downlink. Also, the LLC occupancy is almost orthogonal to the demand of the vBS, yielding an increase of a $2-6\%$ when the demand goes from 10% to 100%.

Figure 3-7 depicts the computing usage of a vBS as a function of the L3 cache ways when we deploy it using 3 cores. Figure 3-8 depicts the computing usage for the different Modulation and Coding Schemes (MCSs) with a low traffic demand (i.e. 20% of the total demand) and high traffic demand (i.e. 100% of the total demand) for uplink and downlink. We observe that there are significant differences between the achievable computing usage reduction. For high traffic both in UL and DL, we can achieve a more significant computing usage reduction. On the contrary, a vBS that processes a low traffic demand can attain lower gains in terms of computing usage. We conclude that LLC resources have different utility depending on the vBS context.

This contrasts with Figure 3-6 which shows that the vBS makes full use of the cache memory regardless of the demand. Thus, if there is no LLC cache allocation mechanism, all vBSs deployed will be using the same amount of LLC cache memory on average. It is key to strategically distribute the LLC cache ways among the vBSs deployed boosting its utility to minimize the computing usage and therefore the energy consumption depending on the traffic demands.

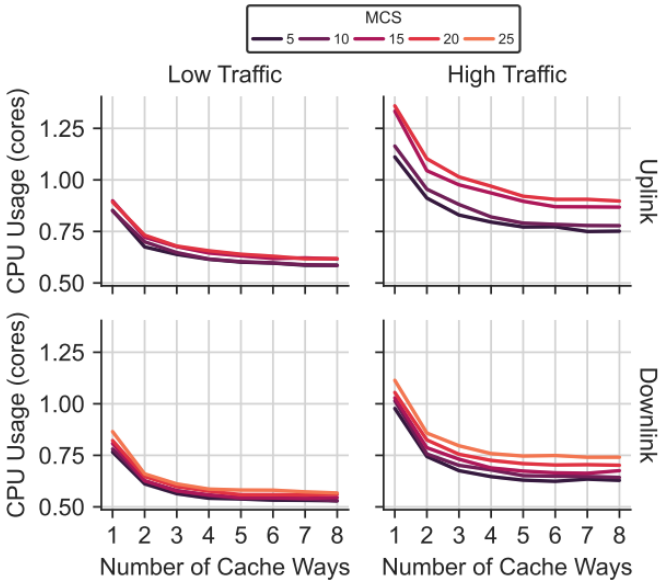


Figure 3-7: vRAN resource allocation - Computing usage as a function of the LLC allocated cache ways for different SNR and in UL and DL.

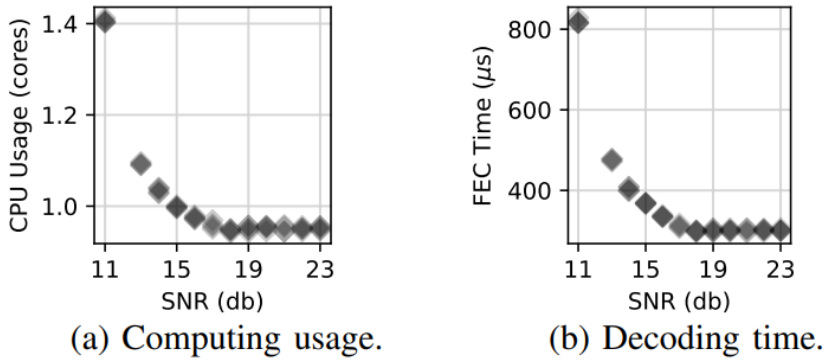


Figure 3-8: vRAN resource allocation - Computing usage and decoding time of a vBS with max. UL and DL load with mild MCS over different SNR conditions.

Quantifying the LLC cache memory utility is a challenging task. The LLC cache utility depends on the computing demands of the vBS which are influenced by various factors, including traffic demand in both the DL and UL, the signal-to-noise ratio (SNR) of each wireless link, and the specific MCS utilized for communication. All these elements interact in a complex manner [14], [6]. Figure 3-8a depicts the relative mean core usage of the vBS, and shows that, given a MCS, lower SNR regimes demand a higher amount of computing resources. The underlying reason is the iterative nature of the forward-error correction (FEC) algorithms – signals received with lower SNR require a higher number of FEC iterations to decode the transported codeword successfully. This is confirmed by Figure 3-8b, which shows the amount of time taken by the decoder to finish its task for every transport block.

3.1.3 Problem formulation

We consider a vRAN platform comprising M_{cores} computing cores and N_{LLC} LLC cache ways. We consider that N_{vBS} vBS instances are deployed in the platform. Every vBS i in the vRAN platform has both a UL and DL traffic demand denoted by d_i^{UL} and d_i^{DL}, respectively; a SNR s_i; and an MCS in UL and DL denoted by m_i^{UL} and m_i^{DL}, respectively, that the radio scheduler selects depending on s_i. The vBS i has a set of isolated cores P_i such that $|P_i| > 0$ and a number of dedicated LLC cache ways n_i^{LLC}, where n $n_i^{\text{LLC}} \geq 1 \; \forall i$. We need to satisfy that $\sum_i P_i \leq M_{\text{cores}}$ and $\sum_i n_i^{\text{LLC}} = N_{\text{LLC}}$. We define c_i as the computing use of vBS i. We define $\boldsymbol{x}_i :=$

$(d_i^{\mathrm{UL}}, d_i^{\mathrm{DL}}, s_i, m_i^{\mathrm{UL}}, m_i^{\mathrm{DL}})$ as the context of the vBS i. Moreover, we define f_i as the function that maps $\boldsymbol{x}_i$ and n_i^{LLC} to the computing usage c_i.

Finally, we define vector χ which concatenates the context vectors from all vBS as $\chi \coloneqq (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{N_{\mathrm{vBS}}})$. Also, we define $\mathcal{P} \coloneqq (P_1, \ldots, P_{N_{\mathrm{vBS}}})$ and $\mathcal{N} \coloneqq (n_1^{\mathrm{LLC}}, \ldots, n_{N_{\mathrm{vBS}}}^{\mathrm{LLC}})$ as the vectors with the core set allocation and the LLC cache ways allocation on a vRAN platform. Similarly, we define the function f which maps $(\chi, \mathcal{N})$ to the total computing usage $C^{\mathrm{vRAN}} \in [0, M_{\mathrm{cores}}]$ of the vRAN platform. We define the problem of optimizing the computing set and the LLC allocation as:

$$\min_{\mathcal{N}} f(\chi, \mathcal{N})$$

$$\text{subject to} \sum_i n_i^{\mathrm{LLC}} = N_{\mathrm{LLC}}$$

As explained in previous deliverables, the computing usage and the energy consumption are proportional. Thus, minimizing the computing usage function f also minimizes the total energy consumption. Note that, in our problem, the assignment of CPU cores to vBS $\mathcal{P}$ is already given. We select a fixed value of $\mathcal{P}$ based on our experimental insights and previous works on this topic as presented in the previous deliverables (see D4.2 [2] for more details). Note that, in the problem presented, the optimal LLC cache allocation depends on the context χ, whose dimensionality increases depending on the number of vBSs. Moreover, the optimal action is also dependent on the number of active vBS, which may change over time. To avoid the exploration burden of learning algorithms (e.g., reinforcement learning) that can lead to suboptimal configurations, we decompose the problem and use a digital twin (DT) system.

3.1.4 MemorAI

To solve the presented problem, we propose an optimization framework which we call MemorAI. This framework considers discrete decision intervals denoted by $t \in \{1, 2, \ldots, T\}$. At the beginning of each decision interval, our solution receives $\chi^{(t)}$, and decides the optimal LLC allocation $\mathcal{N}^{*(t)}$ across all vBSs, to minimize the energy consumption. MemorAI is composed of a set of DTs and a NN classifier. Figure 3-9 shows our optimization framework.

Thanks to full vBS isolation via pinning it to dedicated cores and LLC cache ways allocation, we make the observation that $C^{\mathrm{vRAN}} = \sum_i c_i$ as there are no joint effects between vBSs. This observation implies that $f(\chi, \mathcal{N}) = \sum_i f(\boldsymbol{x}_i, n_i^{\mathrm{LLC}})$. As there is no interaction among vBS in terms of computing usage, we can create DTs of independent vBS. Thus, we can mirror their behavior in a safe and controlled environment for testing and learning. Each DT can model the particularities of each vBS (e.g., different implementations protocol stacks) and we can emulate the complex interactions of the full vRAN system. We create a DT using operational data from one vBSs. Using each vBS's DTs, we can lower the time cost to generate a dataset for a number of vBS as we aggregate the results of each DT. Note that without the DTs, the amount of data needed to create a training dataset increases exponentially with the number of vBS (curse of dimensionality).

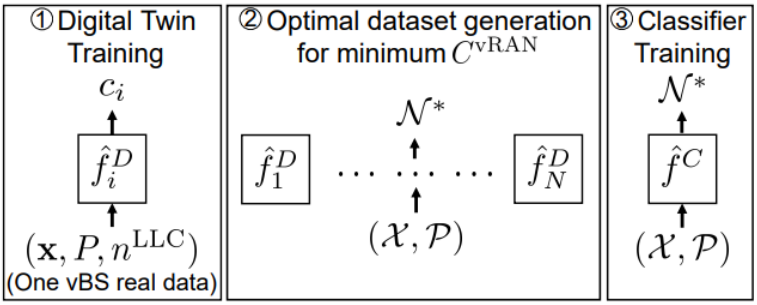


Figure 3-9: vRAN resource allocation - MemorAI optimization framework.

To create the DT of one vBS we used our vRAN platform to generate a training dataset with a fixed computing core set $|P|$. Note that we select a set of cores such that vBS can correctly operate. This makes f_i a continuous function. Each dataset sample contains a 4-tuple with $(\boldsymbol{x}, P, n^{\mathrm{LLC}}, c)$. Using this dataset, we built up a DT using a NN which approximates c using $(\boldsymbol{x}, P, n^{\mathrm{LLC}})$ i.e. it approximates f_i minimizing the Minimum Squared Error (MSE). We denote the DT function as $\hat{f}_i^{\mathrm{D}}$. Figure 3-9, also shows the DT training step (1).

The DT allows us to evaluate the CPU usage of different configurations very accurately without having to use the real system. Therefore, we can perform an exhaustive search to find the optimal LLC allocation $\mathcal{N}^*$ for a set of contexts $(\chi, \mathcal{P})$. Note that the size of the set of possible LLC cache allocations is $|N| = \binom{N_{\mathrm{LLC}}-1}{N_{\mathrm{vBS}}-1}$. Figure 3-9, shows how using the different DTs we generate the previous data set (2). Using this data set, we train a fully connected NN to predict $\mathcal{N}^*$ for a given context $(\chi, \mathcal{P})$. Specifically, we solve a multi-class classification problem using the cross-entropy loss function. We denote the classifier function as $\hat{f}^{\mathrm{C}}$. Note that our solution is very flexible to changes in the system (e.g., upgrades in the implementation of the software stack, deployment of new vBSs, etc.). In those cases, after having the DT modeling, we can easily retrain the NN classifier offline without degrading the performance of the system. Finally, Figure 3-9, shows the classifier training step of our optimization framework (3).

3.1.5 Performance Evaluation

In this section, we evaluate the performance and potential savings of our approach. We carry out the evaluation for a $N_{\mathrm{vBS}} = 5$ vBS deployment. We used PyTorch[9] to implement the DT and the Classifier.

Training Evaluation

1. Digital Twin (DT): We implemented the DT of one vBS using a NN with three hidden layers of sizes $\{256, 128, 64\}$ respectively with a ReLU activation function. Moreover, we stop the training iterations using an early stopping mechanism to prevent overfitting. The early stopping mechanism stops the training after the value of the loss function in a validation data set has not improved during the last N training iterations. N is usually referred to as patience. Figure 3-10 shows the MSE loss value on the training and validation data sets of the DT as a function of the training iterations. We selected a patience of $N = 10$ and trained the model during 70 iterations.
2. NN Classifier: On the other hand, we implemented the Classifier as a NN with 4 hidden fully connected layers of sizes $\{512, 384, 384, 512\}$ respectively. Each layer also used a ReLU activation function and we introduced a 0.2 dropout probability during training. We also use the early stopping mechanism with $N = 50$ of patience. Figure 3-11 shows the cross-entropy loss on the training and validation data set as a function of the number of training iterations. We train the model until iteration 300 (due to the early stopping mechanism). The training loss is higher than the validation loss due to the dropout layers. Also, the classifier achieves a 92.1% accuracy on our testing data set.

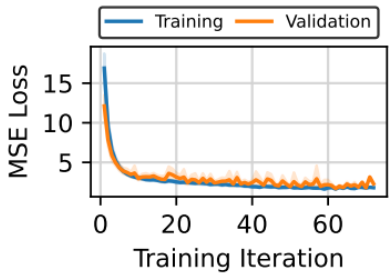


Figure 3-10: vRAN resource allocation - Digital Twin MSE

[9] https://pytorch.org

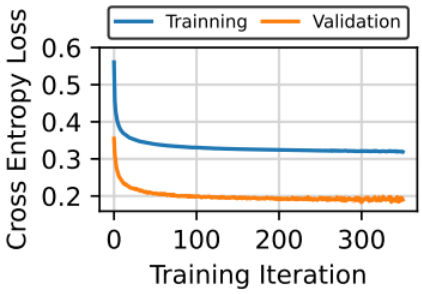


Figure 3-11: vRAN resource allocation - NN Classifier Cross

Performance Benchmark

We design MemorAI to operate in the Non-Real Time RIC of the O-RAN architecture as an rApp. Based on this, we selected a time granularity of 15 minutes for our evaluation. We generated a data set with random context data and compared MemorAI against the following approaches:

- **Random**: We select the allocation of cache ways for each vBS randomly;
- **Equal partition**: All the vBSs get allocated the same number of cache ways. Extra cache ways are left unallocated;
- **Weighted**: Each vBS gets allocated a number of cache ways as proportionally to its total demand as:

$$n_i^{\mathrm{LLC}} = \frac{d_i^{\mathrm{UL}} + d_i^{\mathrm{DL}}}{\sum_j d_j^{\mathrm{UL}} + d_j^{\mathrm{DL}}} \cdot N_{\mathrm{LLC}}$$

Figure 3-12a shows the energy savings in kilojoules (kJ) of our solution and the optimal strategy with respect to the different benchmarks in a decision interval of 15 minutes. Our solution outperforms the three benchmarks in terms of energy while producing almost the same results as the optimal solution. Thus, MemorAI understands better the utility of LLC cache partitioning than benchmark strategies. Our solution yields higher savings up to 1 kJ compared to the random strategy, up to 0.35 kJ compared to the equal partitioning, and up to 0.36 kJ compared to the weighted strategy. Note that, although the weighted strategy shows a lower power consumption, it does not scale the LLC cache properly when there is an imbalance between UL and DL demands. In these cases, our approach achieves the highest gains with respect to this strategy. Finally, Figure 3-12b shows the attained gains when the system has only 8 cache ways available. In that case, the random approach performs better because the number of configurations is lower and therefore the probability of selecting the optimal configuration is higher. The savings compared to the equal partition approach are higher as the benchmark can only allocate one cache way per vBS in this scenario. Compared to the weighted strategy, we still observe higher savings.

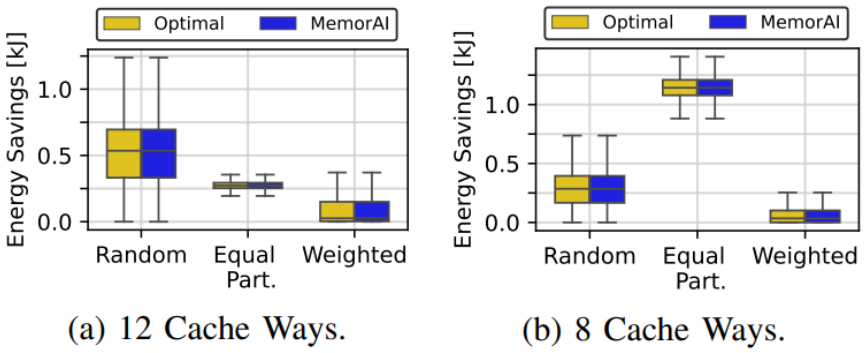


Figure 3-12: vRAN resource allocation - Energy savings compared to different benchmarks and different number of cache ways for a 15 min decision interval

3.1.6 Conclusions

Cache memory is a key resource for vRANs to reduce energy consumption. Non-isolated access to cache memory resources increases energy consumption, making less attractive the advantages of virtualization. In this final deliverable, we have studied how the different mechanisms for cache memory isolation decrease the energy consumption of a vRAN platform. Then, we proposed MemorAI which strategically allocates LLC resources to minimize energy consumption. MemorAI comprises a digital twin and a NN classifier, providing a very efficient and flexible solution. MemorAI achieves almost optimal performance and can attain significant energy savings when compared with other strategies.

3.2 AI/ML and data-driven strategies for energy-efficient 5G carrier on/off switching

In this section, we present an AI-driven QoS-aware energy-saving strategy to manage 5G carrier on/off switching. The objective is to enhance energy efficiency while ensuring a certain throughput level during switch-off periods. This work is aligned with BeGREEN D2.3 [19] (Section 4.8), where we evaluated existing switch-off opportunities in a real cellular deployment by using data from a commercial Mobile Network Operator (MNO). The approach involved offloading 5G cell traffic to the 4G cells of the same site and sector.

Recall that the optimal strategy, i.e., an oracle-like approach with full dataset visibility, quantified potential switch-off opportunities across the 200 evaluated cells. However, the achievable energy savings decreased when specific QoS requirements were imposed. Specifically, the estimated Energy Savings dropped from 79% of total consumption (13.7 MWh) to 17% (2.9 MWh) when increasing the throughput constraint from 0 Mbps (no constraints) to 25 Mbps (high constraints). Figure 3-13 illustrates this trade-off, showing a heat map of energy-saving opportunities (left) and the energy savings vs. QoS trade-off (right). Achievable throughput levels were estimated according to the correlation between cell load and average UE throughput level found in the dataset, as analysed BeGREEN D2.3 [19]. The considered levels were selected according to typical throughputs required by video streaming applications, such as Netflix or YouTube.

In BeGREEN D4.2 [2] we defined general energy-saving strategies, including both heuristics and ML-based, but those did not contemplate QoS aspects. To develop a QoS-aware strategy, as mentioned earlier in Section 2.3.1, we developed and evaluated Logistic Regression classifiers. To reduce the computational cost of analysis and optimization, the study was limited to a subset of 70 cells located in an urban scenario characterized by a high diversity of switch-off opportunities. Figure 3-14 illustrates the selected cell set and the distribution of switch-off times over the course of a full week.

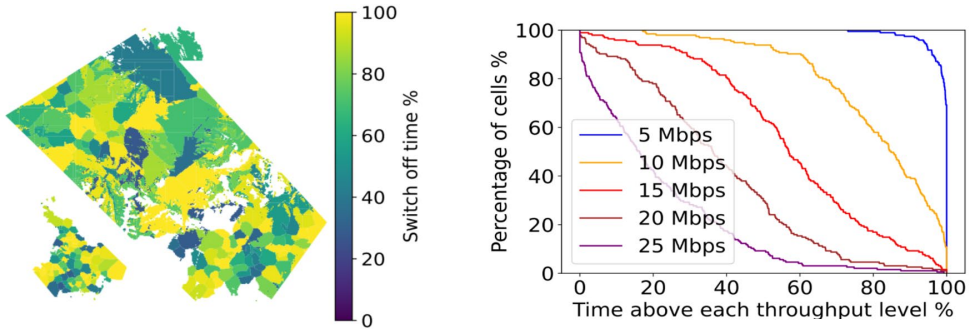


Figure 3-13: 5G Carrier on/off switching – Energy Savings opportunities (left), and Energy Savings/QoS trade-off (right).

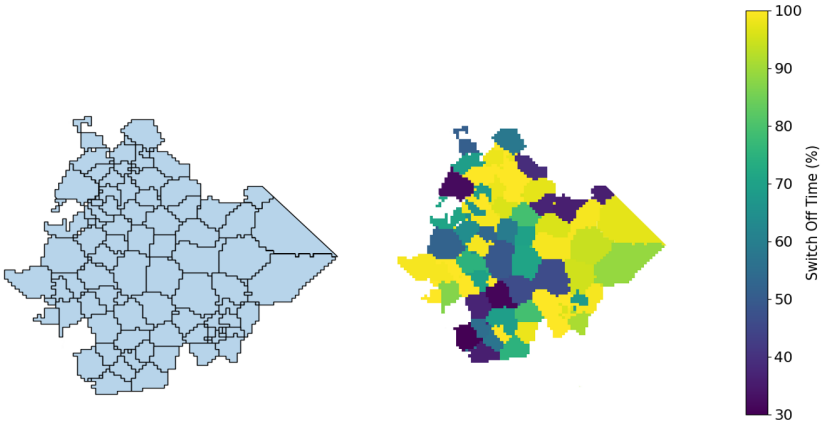


Figure 3-14: 5G Carrier on/off switching – Reduced set of cells (left), and Energy Savings opportunities (right).

3.2.1 Classifier description

To train the model, we generated a dataset containing information on switch-off opportunities and the QoS provided during switch-off periods. This was achieved by collecting results from the optimal strategy applied over two weeks of data across all cells in the deployment, with an additional week reserved for initial evaluation and hyperparameter tuning, and a fourth week for evaluation of the different policies. Recall that the oracle-like strategy based its QoS calculations on a regression method based on historical cell-level data. Our QoS estimation method was found to have a mean absolute error of 4.7 when evaluated over an unseen week of data across all dataset cells. Therefore, we decided to separate output columns in discrete intervals of 5 Mbps.

The classifier takes as input the 5G load and the load of active 4G cells within the same site and sector. It outputs six binary decisions, each corresponding to a predefined throughput level: 0 Mbps, 5 Mbps, 10 Mbps, 15 Mbps, 20 Mbps, and 25 Mbps. The logic follows the same approach used in BeGREEN D4.2 [2] for the simple classifier strategy but extends it to all considered throughput levels. Each output column indicates, for each time period, whether a given 5G cell can be switched off while still meeting the throughput requirements of the 4G cells used for offloading. When deploying the model, two types of erroneous decisions may occur:

- Missed energy-saving opportunities: The model decides to keep the 5G cell ON, even though the traffic could have been offloaded and the throughput requirement met.
- Outage decisions: The model decides to switch the 5G cell OFF, but the throughput requirement cannot be met, or the traffic cannot be offloaded at all (i.e., a 0 Mbps case).

The missed opportunities decisions lead to wasted energy. However, note that for an operator those are not as critical as the outage decisions, which can make UEs to experience lower throughputs than the required ones or lead to cell saturation. This trade-off between missed energy-savings and outage decisions can be weighted during model training by tuning the “class weight” hyperparameter. It allows setting specific weights to each of the output classes, which are 0 (or OFF), and 1 (or ON), so that the model prioritizes the learning of one class over the other according to the weight’s ratio.

We evaluate the effect of tuning the class weights on the recall and precision metrics obtained over the training set, which are directly linked with the outage and missed opportunities decisions. Those are defined as:

$$Recall = TP \div (TP + FN)$$
$$Precision = TP \div (TP + FP)$$

where TP, FN, and FP are True Positives, False Negatives, and False Positives, respectively

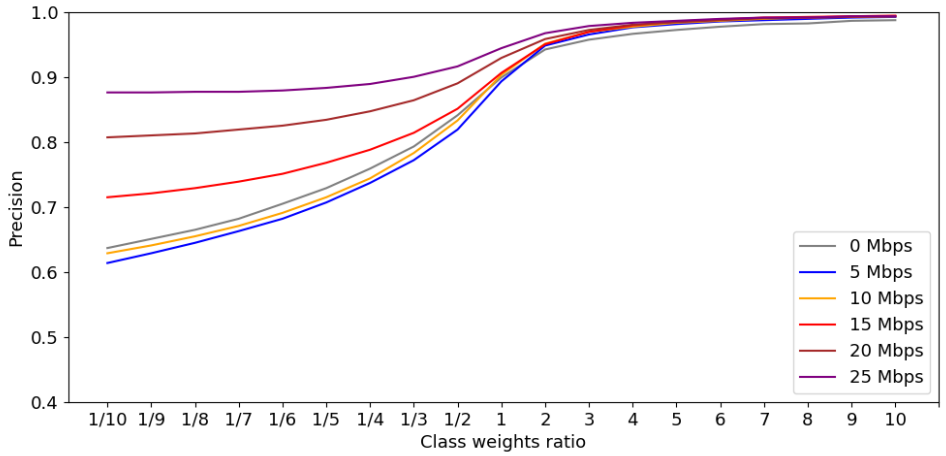


Figure 3-15: 5G Carrier on/off switching – Precision curve for the Logistic Regression multi-output classifier.

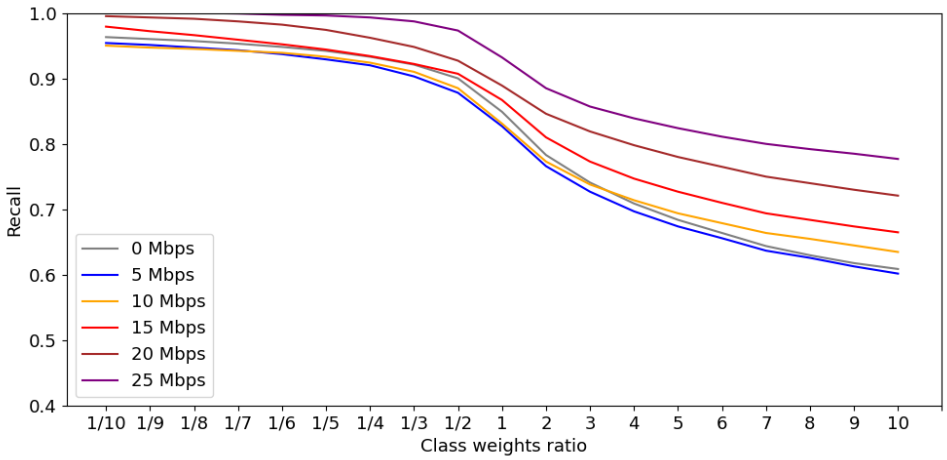


Figure 3-16: 5G Carrier on/off switching – Recall curve for the Logistic Regression multi-output classifier.

The outage decisions, or False Negatives, get reduced in the high recall domain. Conversely, the missed opportunity decisions, or False Positives, get reduced in the high precision domain. To get an intuition on how the class weights affect the model, and the trade-off between precision and recall, we evaluated our model for different class ratios.

Figure 3-15 and Figure 3-16 show the obtained results. We observe that when the class ratio is below one, recall is increased while precision decreases; conversely, when the ratio is above one, precision improves at the expense of recall. To reduce outage decisions, it is preferable to use class ratios below one. However, very low ratios significantly reduce precision, leading to an increase in missed energy-saving opportunities. We also appreciate some differences between the output columns' behaviour (i.e., QoS levels), which are related to the distribution of on/off decisions in each column Higher throughput levels present fewer opportunities for switching off, which leads to a faster increase in recall for those cases.

The differences observed between output columns may motivate the use of individual hyperparameter tuning to optimize each column's performance. Note that the multi-output classifier used is an ensemble of estimators, with each estimator trained to fit a specific output column, effectively functioning as six individual classifiers. However, hyperparameters are set globally, meaning all output columns share the same configuration and setting individual weights are not allowed. Conversely, compared to the training of six independent classifiers for each of the QoS levels, this approach offers significant advantages in terms of computational cost, training and retraining speed, energy consumption, and inference efficiency.

To assess the impact of this trade-off, the next section compares results obtained with the multi-output

classifier against those from six independently trained classifiers, where individual hyperparameter tuning is possible.

3.2.2 Classifier evaluation

In this section, we present the results obtained over the evaluation week, focusing on two main aspects: the effect of class weighting, and the comparison between multi-output and single-output classifiers (i.e., global versus individual class weight settings). Since our objective is to prioritize the reduction of outage decisions, we evaluate class weight ratios below 1, which maximize recall metric. Next figures illustrate the average number of outage decisions (Figure 3-17), missed opportunities (Figure 3-18), and total erroneous decisions (Figure 3-19) across cells as the class ratio decreases. The outage decisions and the missed opportunities figures are normalized to the total amount of ON and OFF decisions, respectively, while the erroneous decisions figure is normalized to the total amount of decisions taken by the classifier.

Results in the figures reveal that, as the class ratio decreases, missed opportunities increase more rapidly than the reduction in outage decisions. This trend is confirmed in the total erroneous decisions plot, which consistently shows an increase with the class ratio. One of the key takeaways from this analysis is that reducing outage decisions inevitably leads to a disproportionately greater rise in missed energy saving opportunities, as can be inferred from the impact of lowering the class weights ratio on precision metric show in Figure 3-15. Note that in the case of 20 and 25 Mbps, the total number of erroneous decisions, illustrated in Figure 3-19, tends to stabilize with the increase of the weights. However, this is caused by the very reduced number of OFF decisions being taken for these QoS levels, as was shown in Figure 3-18, which consequently leads to almost zero outage decisions, as depicted in Figure 3-17.

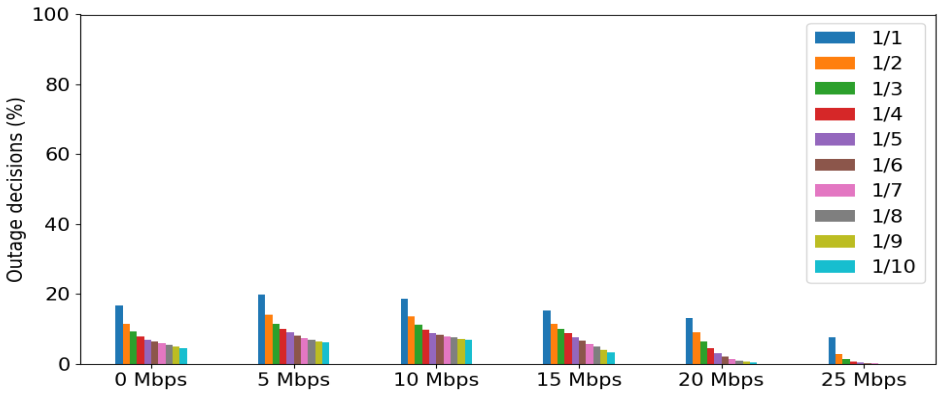


Figure 3-17: 5G Carrier on/off switching – SLA Outage decisions according to the class ratio

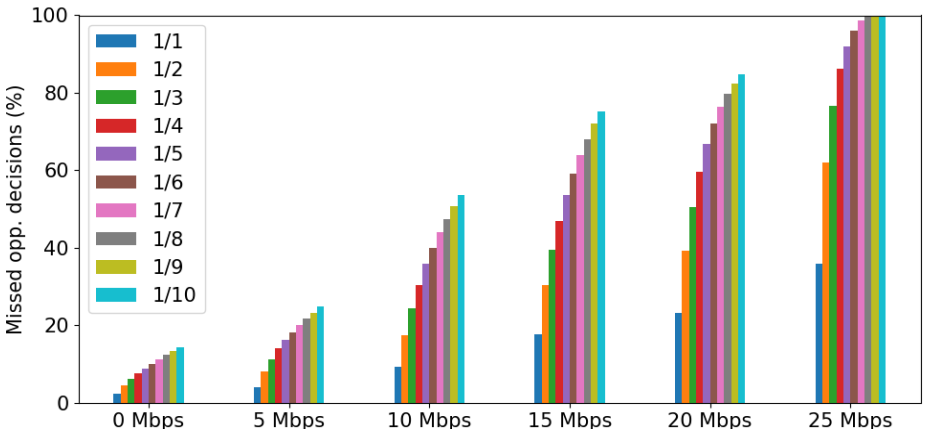


Figure 3-18: 5G Carrier on/off switching – Missed energy saving opportunities according to the class ratio

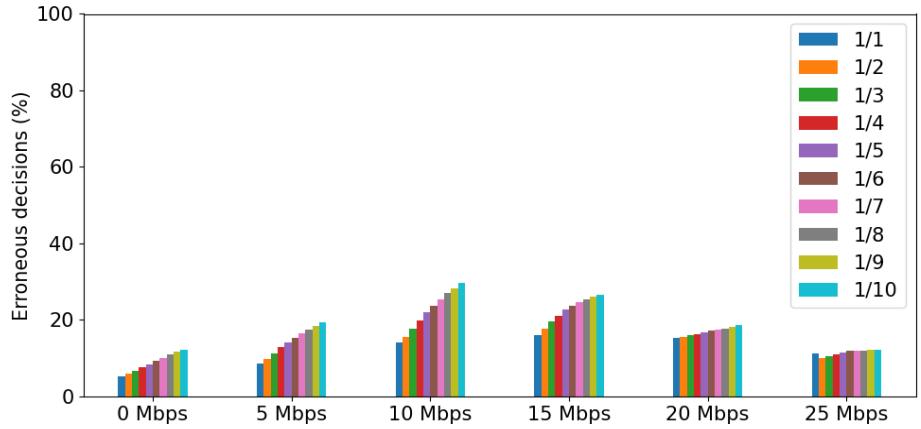


Figure 3-19: 5G Carrier on/off switching – Total erroneous decisions according to the class ratio

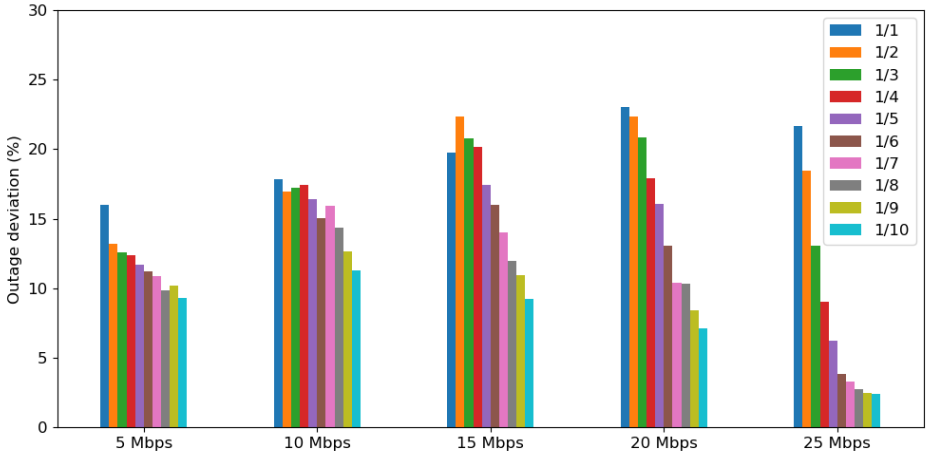


Figure 3-20: 5G Carrier on/off switching – SLA Outage deviation according to the class ratio

Figure 3-20 quantifies the impact of Service Level Agreement (SLA) outages on expected throughput. The metric is expressed as a percentage relative to each SLA level: for example, a 10% deviation in the 5 Mbps column corresponds to an outage of 0.5 Mbps or less, while the same percentage in the 25 Mbps column represents an outage of 2.5 Mbps or less. Once again, adopting a more conservative approach clearly improves this metric, especially in the case of high QoS levels, but at the cost of reducing switch-off opportunities.

The observed trends, both in decreases and increases, vary between columns, with some exhibiting significantly larger errors or achieving more substantial reductions in outage decisions than others. This supports the hypothesis that individual hyperparameter tuning for each column may lead to improved overall performance. To investigate this, we compare the performance and energy saving results obtained using the multi-output classifier versus six independently trained single-output classifiers, under two different policy constraints: (i) Outage decisions must remain below 10% and (ii) outage deviation must remain below 15%.

The key difference between both approaches lies on how the class weight ratio is determined. In the case of the multi-output classifier, a single global ratio must be selected and applied, which ensures that the policy constraint is met for every column. In contrast, when using six individual classifiers, each column can be trained with its own specific class ratio: i.e., the least restrictive one that still satisfies the policy based on the results obtained in prior evaluations.

Following this approach, for the first policy constraint (outage decisions must remain below 10%), we train the multi-output classifier with a uniform class ratio of 1:4 across all columns.

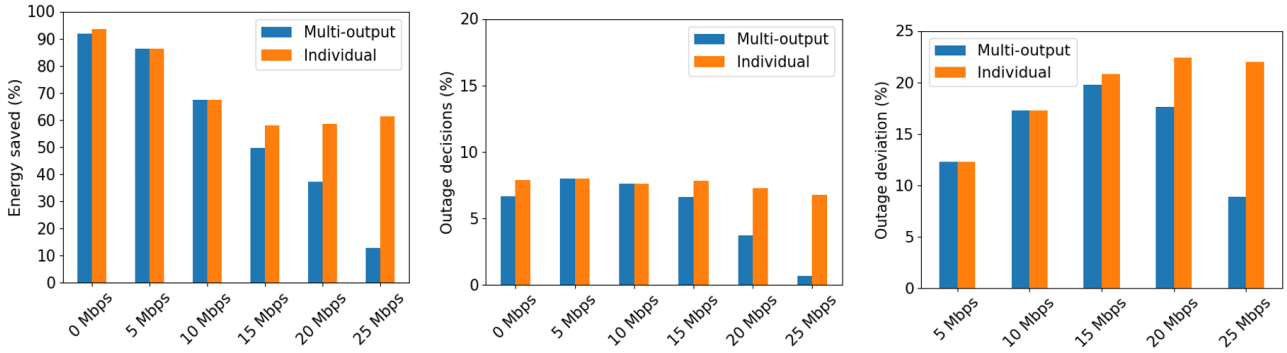


Figure 3-21: 5G Carrier on/off switching – Performance of classifiers with selected class ratios (Outage decisions below 5%)

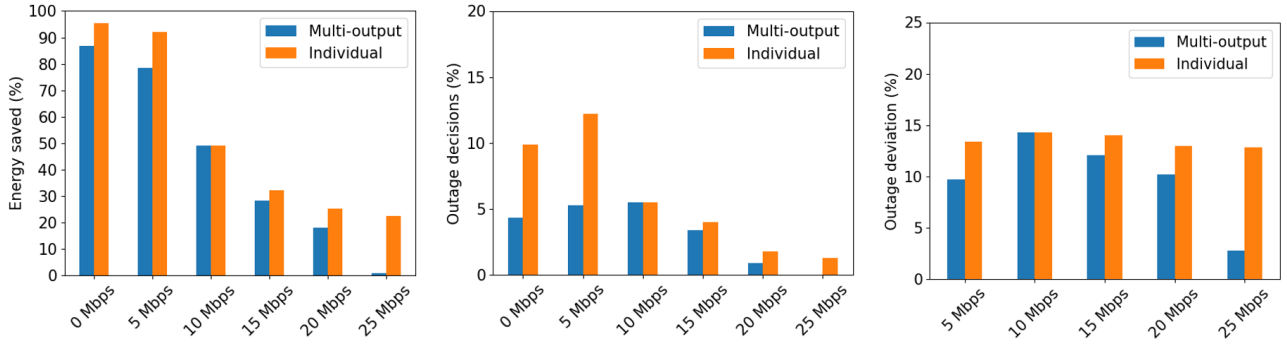


Figure 3-22: 5G Carrier on/off switching – Performance of classifiers with selected class ratios (Outage deviation below 15%)

In contrast, the individual classifiers are tuned with column-specific ratios of 1:3, 1:4, 1:4, 1:3, 1:2, and 1:1, respectively for each QoS limit. Figure 3-21 shows the results obtained for the first policy. The left figure illustrates the percentage of energy saved relative to the oracle-like strategy, the centre figure shows the total number of outage decisions, and the right figure the outage deviation. As previously discussed in the context of the recall/precision trade-off, lower class ratios have a more significant impact on high-throughput levels, as these offer fewer switch-off opportunities. Therefore, individual classifiers outperform multi-output classifier in the amount of energy saved, since they are allowed to use less restrictive class ratios while still meeting policy constraints (as shown in the centre figure). In contrast, the multi-output model results in fewer outage decisions and deviations, as it requires a more conservative class ratio. However, the individual classifiers in some cases can reach outage deviations of around 20%, which may be problematic in certain scenarios.

Consequently, the second policy considers the outage deviation, trying to decrease trying to decrease the difference between the obtained and the SLA throughputs. The multi-output class ratio is therefore set to 1:8, while individual classifiers are configured to 1:2, 1:2, 1:8, 1:7, 1:6 and 1:3, respectively for each QoS limit. Since the 0 Mbps outage decisions translate into saturation of the 4G cells (i.e., no measurable outage deviation exists) in the case of individual classifiers we impose the same ratio as in the 5 Mbps column. Note that, compared to the previous case, this is a most restrictive policy for almost all QoS constraints (except 0 and 5 Mbps cases) leading in the other cases to less energy savings and outage decisions. Also, the impact of the common class weight in the multi-output classifier is particularly severe for the highest QoS constraint, resulting in almost no energy savings. These results highlight the advantages of individually tuned classifiers.

Once it was validated that tuning class weights in individual classifiers can outperform multi-output models, we extended the comparison to a different type of classifier. Specifically, we tested a widely known tree-based model: the XGBoost Classifier. This model uses a tree-boosting algorithm that has proven effective in classification tasks, even when handling missing data.

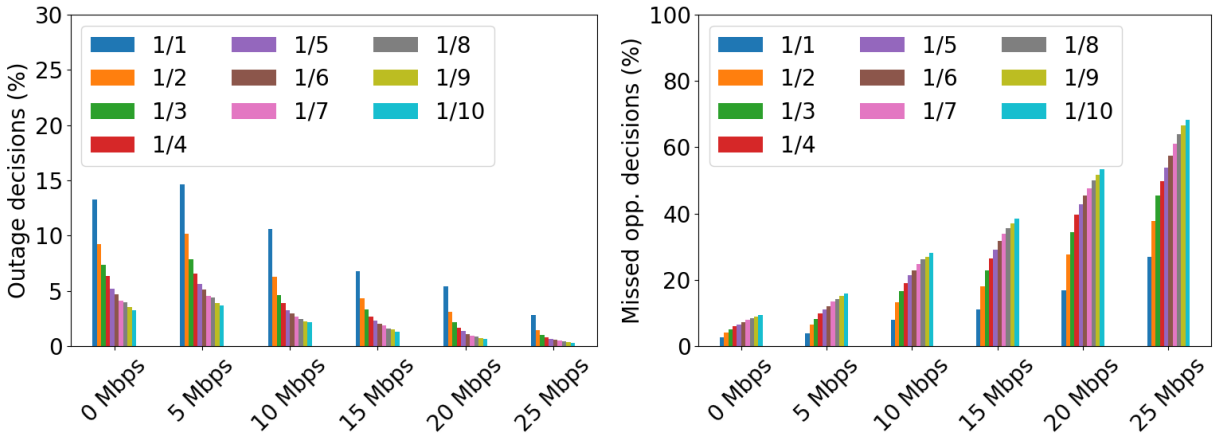


Figure 3-23: 5G Carrier on/off switching – SLA Outage decisions and missed opportunities according to the class ratio (XGBoost)

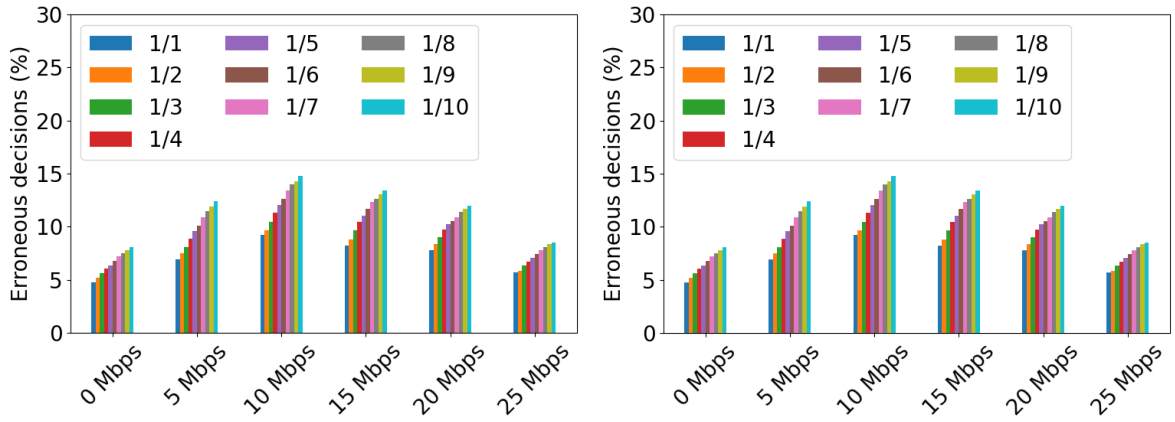


Figure 3-24: 5G Carrier on/off switching – Erroneous decisions and outage deviation according to the class ratio (XGBoost)

We followed the same methodology as with the Logistic Regression classifier. First, we use a testing week to analyze the trade-off between outage decisions and missed opportunities. Then, under identical SLA policies or constraints, we determined the optimal class ratios for the evaluation week.

Figure 3-23 and Figure 3-24 show the results obtained for the testing week. As illustrated, the XGBoost Classifier achieves better performance than Logistic Regression. It results in significantly fewer outage decisions, and missed opportunities increase more gradually. These outcomes suggest that the XGBoost Classifier is more suitable for the targeted use case.

We next analyzed the results obtained under the same policies or SLA constraints considered in the previous section: an outage decision rate below 10% and an outage deviation below 15%. For the XGBoost Classifier, the corresponding class ratios for the first policy were: 1/2, 1/3, 1/2, 1/1, 1/1, 1/1. For the second policy, the ratios were: 1/3, 1/3, 1/1, 1/1, 1/3, 1/1. It is worth noting that, in general, XGBoost allows for more aggressive weight selection, which results in a lower number of missed opportunities. Figure 3-25 and Figure 3-26 present a comparison between the individual classifier approach using Logistic Regression and XGBoost, with the respective computed class weights for both policies.

Figure 3-25: 5G Carrier on/off switching – Comparison between Logistic Regression and XGBoost Classifier on policy 1 (outage decisions < 10%)

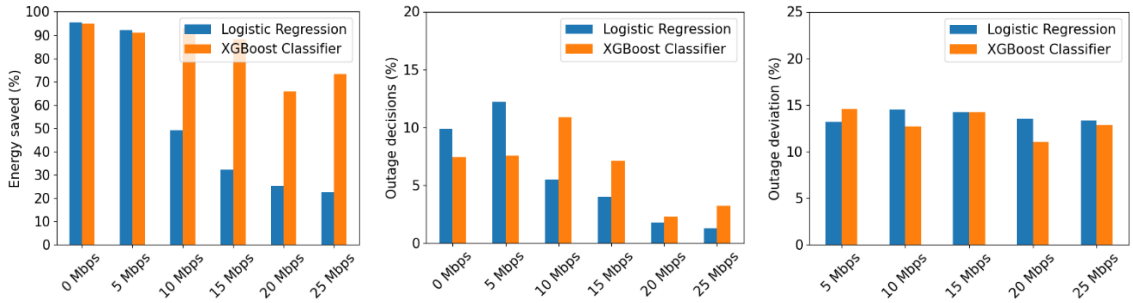


Figure 3-26: 5G Carrier on/off switching – Comparison between Logistic Regression and XGBoost Classifier on policy 2 (outage deviation < 15%)

Except for the outage decisions at 0 Mbps and the outage deviation at 5 Mbps, the XGBoost Classifier outperforms the Logistic Regression model under the first policy, demonstrating significant improvements in both energy savings and reduced outage decisions. This effect is especially pronounced under high-throughput SLA conditions, suggesting that XGBoost handles unbalanced class distributions more effectively.

The performance gap is even wider under the second policy. The stricter constraints imposed by this policy force the Logistic Regression model to adopt much lower class ratios, while the XGBoost Classifier can consider more aggressive weighting. As a result, XGBoost achieves higher energy savings at high throughput levels, while maintaining outage deviations below the 15% threshold. However, Logistic Regression is more conservative in its outage decisions, mainly due to the lower class weights required to comply with the policy.

3.2.3 Conclusions

Energy-efficient cell control through dynamic cell on/off switching is a key strategy for achieving energy savings in cellular networks and represents one of the primary energy-efficiency use cases within O-RAN. AI/ML plays a crucial role in this context, as accurate load and QoS predictions are essential to enable proactive and automated control loops that improve energy efficiency in the network without compromising traffic performance. Across the various deliverables of this Work Package and BeGREEN D2.3 [19], we have leveraged data from a real 5G NSA deployment provided by a MNO to characterize the trade-off between energy-saving opportunities and QoS impacts when switching off 5G cells and offloading their traffic to 4G cells in the same site and sector.

First, we demonstrated that the strong correlation between cell load and energy consumption, combined with the regularity of day-night traffic cycles, makes this scenario well-suited for the application of predictive algorithms that decide on cell on/off switching based on 4G and 5G load patterns. However, due to the different characteristics of 5G and 4G cells, traffic offloading can sometimes result in QoS degradation. In this deliverable, we incorporate QoS constraints into the decision-making loop, and characterize the trade-offs and performance of both single and multi-output classifiers in making on/off decisions. Results

demonstrate that individually tunning the weights of the models according to QoS constraints can improve decision-making and provide notable energy savings without compromising performance. Finally, we compared the performance of the Logistic Regressor with XGBoost. Findings highlighted that XGBoost is a more suitable option for our use case, enabling substantial energy savings even under more restrictive policy constraints than those considered so far.

Future work will explore the application of these strategies to digital twins emulating realistic 5G networks. This include characterizing their performance according to real-time traffic and UE mobility, and establishing criteria for conflict mitigation, model retraining and parameter tunning.

3.3 AI/ML-based algorithmic solutions for relay-enhanced RAN control

BeGREEN D4.1 [1] and D4.2 [2] presented a description of different algorithmic solutions with the purpose to identify the presence of coverage holes (CH) in cellular networks and mitigate them by means of relays. On the one hand, a coverage hole detection methodology based on a clustering approach was proposed. On the other hand, an algorithm to identify candidate RUEs (i.e. UEs with relaying capabilities) was presented with the objective to assess the potentials of RUEs to mitigate coverage holes. Moreover, a fixed relay placement algorithm was also described. This relay placement algorithm aims to determine adequate locations to place fixed relays in order to address the presence of coverage holes. Finally, a relay activation/deactivation algorithm based on a Deep Q Network (DQN) was proposed with the aim to dynamically activate the relays to serve UEs located inside the coverage hole regions and deactivate them when they are not necessary, with the objective to reduce power consumption. An initial evaluation of the proposed algorithms was presented in D4.2 [2].

This section presents an extension of the initial results provided in BeGREEN D4.2 [2]. The proposed solutions have been evaluated in a realistic scenario using real measurements of the UE space/time distribution. Section 3.3.1 presents a description of the considered scenario. Additionally, a description of the gNB and relay power consumption model is also provided. Then, Section 3.3.2 presents the performance evaluation of the different solutions in terms of the UE perceived spectral efficiency and the power consumption. First, an initial analysis in which no relays are deployed is presented, and a characterization of the identified coverage holes is provided. Then, a comparison of a solution based on the use of RUEs to mitigate the coverage holes is compared with respect to the case of no relays and of deploying fixed relays. Finally, the power consumption of different relay-based solutions is provided. In particular, a solution based on deploying fixed relays that remain always active, a solution that deactivates fixed relays when they are not necessary, and a solution based on RUEs are evaluated and compared with respect to the benchmark case of no relays.

3.3.1 Considered scenario

The proposed algorithms have been evaluated in a realistic scenario in a University Campus of the *Universitat Politècnica de Catalunya* (UPC). The considered region is a 325m x 125m area with 25 buildings of 3 floors, denoted as A1, A2, …, D6, as shown in Figure 3-27. 5G NR coverage on the Campus is provided by three outdoor gNBs of a public MNO. This scenario has been modelled by means of a system level simulator that considers the geographical locations of the different buildings and the propagation loss of the transmitted signal. The Urban Macro (UMa) propagation model described in [31] is considered for the link between the gNBs and the fixed relays, RUEs and UEs. For the link between the indoor fixed relays and UEs, the Indoor Hotspot (InH) propagation model from [31] is used. The propagation model in the link between the RUEs and UEs is taken from [32].

The propagation models include outdoor-to-indoor losses and 2D spatially correlated shadowing.

Google. Imagery @2024 CNES/Airbus, Institut Cartogràfic de Catalunya, Maxar Technologies, Map data @2024, Spain.

Figure 3-27: Relay-enhanced RAN control - Considered scenario.

The space/time UE location characterization is based on real measurements of the time evolution of the number of users located in the different buildings obtained from dataset [33]. Specifically, the dataset contains real measurements of the number of users connected in the area of each Wi-Fi APs at the university campus facilities obtained for multiple hours of different days. Thus, it reflects a realistic space/time UE location distribution that is taken as input for the 5G system level simulator assuming the UEs are connected to the gNBs. The considered algorithms have been evaluated using a collection of $D=21$ days of measurements. The clustering process executed in the Coverage hole detection algorithm has been done in $N=24$ time periods of $T=1$ hour, for all the days. The rest of parameters of the proposed algorithmic solutions are also presented in Table 3-1. Concerning the candidate RUE identification process, the algorithm searches for candidate RUEs that are located in the same building and floor in which the coverage hole is detected, excluding the coverage hole area. The same search area is considered for the fixed relay placement process.

Table 3-1: Relay-Enhanced RAN Control - Considered Simulation Parameters

Simulation parameters	Value
D	21 days
N	24 periods
T	3600 s
gNB carrier frequency	3.7GHz
Number of Resource Blocks at the gNB (M_{NB})	273
Number of Resource Blocks at the relay (M_R)	51
gNB channel bandwidth ($B_{NB,tot}=M_{NB}\cdot B_{RB,NB}$)	100MHz
gNB max. transmitted power ($P_{NB,max}=P_{RB,NB}\cdot M_{NB}$)	43dBm
gNB transmitted antenna gain	12dB
Path loss model gNB	UMa [31]
Relay/RUE carrier frequency	3.5GHz
Relay/RUE channel bandwidth ($B_{R,tot}=M_R\cdot B_R$)	100MHz
Relay/RUE max. transmit power ($P_{R,max}=P_{RB,R}\cdot M_R$)	5dBm
Relay/RUE antenna gain	3dB
Path loss model fixed relay	InH [31]
Path loss model RUE	[32]
UE antenna gain	3dB
Noise power spectral density	-174dBm/Hz
Noise Figure	9dB
Efficiency factors (ε_{NB}, ε_R)	0.59

To characterize the coverage improvements achieved with the different solutions, the spectral efficiency observed by the UEs in the downlink direction is considered. The spectral efficiency S_D when a UE is directly connected to the gNB with the highest SINR is computed by using the Shannon formula as:

$$S_D = min[S_{max}, log_2(1 + SINR_{NB-UE})] \quad (3\text{-}1)$$

where $SINR_{NB\text{-}UE}$ is the SINR in the link between the gNB and the UE, and S_{max} is the spectral efficiency corresponding to the maximum MCS of 5G NR from [34]. In turn, when the UE is connected to the gNB via a relay, the spectral efficiency is limited by the link with the worst conditions between both the gNB-relay and relay-UE links and it is expressed as:

$$S_R = min\{S_{max}, log_2[1 + min(SINR_{NB-relay}, SINR_{relay-UE})]\} \quad (3\text{-}2)$$

where $SINR_{NB\text{-}relay}$ and $SINR_{relay\text{-}UE}$ denote the SINR in the link between the gNB and the relay and the link between the relay and the UE, respectively. It is assumed that the UEs select their connectivity (i.e. directly to the best serving gNB or through a relay) that maximises their spectral efficiency.

For the evaluation of the power consumption at the gNBs and the relays, the model proposed in [35] is considered, which assumes that the relation between the gBN transmitted power ($P_{T,NB}$) and the gNB power consumption ($P_{C,NB}$) of a specific gNB is nearly linear. Then, the gNB power consumption can be determined as:

$$P_{C,NB} = P_{0,NB} + a_{NB} \cdot P_{T,NB} \quad (3\text{-}3)$$

where $P_{0,NB}$ represents the gNB power consumption at zero RF output power associated to circuits, signal processing, etc., and a_{NB} corresponds to the linear dependency between the total gNB power consumption $P_{C,NB}$ and the transmitted power $P_{T,NB}$. Similarly, the total power consumption at the R relays associated to a specific gNB is:

$$P_{C,NB} = P_{0,NB} + a_{NB} \cdot P_{T,NB} \quad (3\text{-}4)$$

Where $P_{0,R}$ and a_R are the power consumption model parameters for the relays and $P_{T,r}$ is the transmitted power at the r-th relay.
The details of how the transmitted powers $P_{T,NB}$ and $P_{T,r}$ are calculated are described in Appendix 1. Finally, the total power consumption in a specific gNBs and its associated R relays is:

$$P_{C,R} = \sum_{r=1}^{R} (P_{0,R} + a_R \cdot P_{T,r}) \quad (3\text{-}5)$$

Different values of the power consumption model parameters are considered based on [36][37][38]. Table 3-2 shows the considered combinations of parameters.

Table 3-2: Relay-enhanced RAN control - Power consumption model parameters.

Combination	a_{NB}	$P_{0,NB}(W)$	a_R	$P_{0,R}(W)$
C1	28.4	156.38	20.4	13.91
C2			4	6.8
C3	4.7	130	20.4	13.91
C4			4	6.8
C5	2.8	84	20.4	13.91
C6			4	6.8
C7	2.57	12.85	20.4	13.91
C8			4	6.8

The total power consumption is calculated for different CH mitigation solutions described below:

- *Benchmark, No Relays (BNR):* In this case, no relays are deployed in the scenario, and then, all UEs connect directly to the best serving gNB.
- *Deployment of Fixed Relays Always On (FRAO):* In this solution, it is assumed that the fixed relay placement algorithm described in D4.2 [2] has been executed and a fixed relay has been deployed to address each identified CH. In this solution, the fixed relays remain always switched on, even in situations in which they are not serving UEs.
- *Deployment of Fixed Relays On/Off (FROO):* In this solution, the deployment of fixed relays is assumed, and each of the fixed relays is switched off when there are no UEs to be served.
- *Deployment of Fixed Relays and RUEs (FRR):* In this solution, it is assumed that the candidate RUE identification process described in D4.2 is active, and some CHs are addressed by means of RUEs. In case of low availability of RUEs for some specific CHs, fixed relays are deployed according to the proposed fixed relay placement algorithm. As in the FROO solution, fixed relays are switched off when there are no UEs to be served.

For the solutions FRAO, FROO and FRR the Power Saving (PS) with respect to the BNR is determined as the relative difference in percentage between the power consumption of the solution and that of the BNR. It is worth mentioning that a positive power saving (i.e. PS>0) means that the considered solution consumes less power than the BNR benchmark case of no relays, while a negative value (i.e. PS<0) indicates that the case of no relays provides less power consumption.

3.3.2 Performance evaluation

This section evaluates the performance of the proposed solutions in the considered scenario. First, an initial analysis in which no relays are deployed is presented in section 3.3.2.1, including the obtained characterization of the detected coverage holes. Then, section 3.3.2.2 presents the results of the candidate RUE identification process, while section 3.3.2.3 evaluates the performance results obtained with RUEs for different RUE activation policies. Section 3.3.2.4 presents a comparison of the performance for the case when no relays are considered, the case of deploying the fixed relays and a solution based on RUEs. Finally, Section 3.3.2.5 compares the FRAO, FROO and FRR solutions in terms of power saving with respect to BNR.

3.3.2.1 Benchmark – no relays

With the aim of illustrating the overall performance in the network without relays, Figure 3-28 shows the Cumulative Distribution Function (CDF) of the spectral efficiency observed by the UEs when they are directly connected to the best serving gNB. The average spectral efficiency in the whole scenario is around 3.4bits/s/Hz. According to Figure 3-28, in 21% of the measurements reported by the UEs, the maximum spectral efficiency of S_{max}=7.4bits/s/Hz is obtained, which corresponds to geographical regions with very good propagation conditions. In turn, from Figure 3-28 it can also be observed that there is around 10% of UE collected measurements with a spectral efficiency lower than 0.5bit/s/Hz that correspond to measurements collected at specific indoor regions with a very low received signal level in certain times.

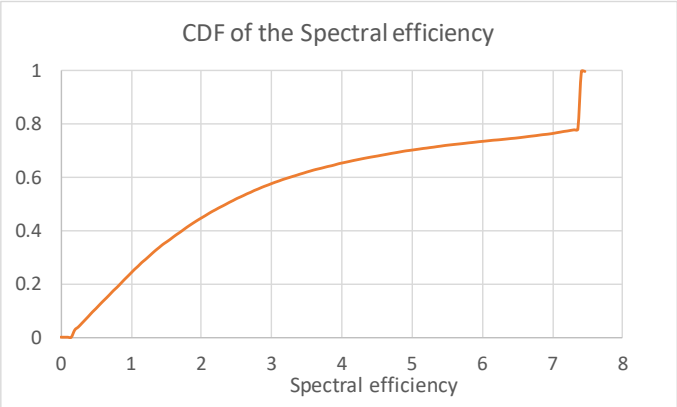


Figure 3-28: Relay-enhanced RAN control - CDF of the spectral efficiency in the indoor locations.

An analysis of the propagation conditions and the spatio-temporal user distribution has been conducted to illustrate the problematic being addressed. Figure 3-29 presents a map with the spectral efficiency in bits/s/Hz obtained at the ground floor of the different buildings. This spectral efficiency is calculated according to (3-1) with the parameters described in Table 3-1 and using the UMa propagation model in [31]. Additionally, Figure 3-30 and Figure 3-31 plot a map of the spatial density, in UEs per square meter, in the ground floor of the different buildings in two different days. This is calculated as the average normalized time in which a UE is located at each geographical location.

The results indicate that the user distribution is non-uniform across the campus. Certain buildings exhibit higher user density than others, and even within the same building, varying user concentrations are observed on different days. As an example, Building A3 (see building names in Figure 3-27) displays different user concentrations in the two days represented in Figure 3-30 and Figure 3-31, while Building B6 consistently exhibits elevated and sustained user density in both days. When considering the user density in these buildings jointly with the spectral efficiency map of the campus, it is observed that the propagation conditions in Building A3 allow users to experience good levels of spectral efficiency (see Figure 3-29). In contrast, users in building B6 are likely to experience significantly lower spectral efficiency values in an area with sustained high traffic levels, which could result in the presence of a CH.

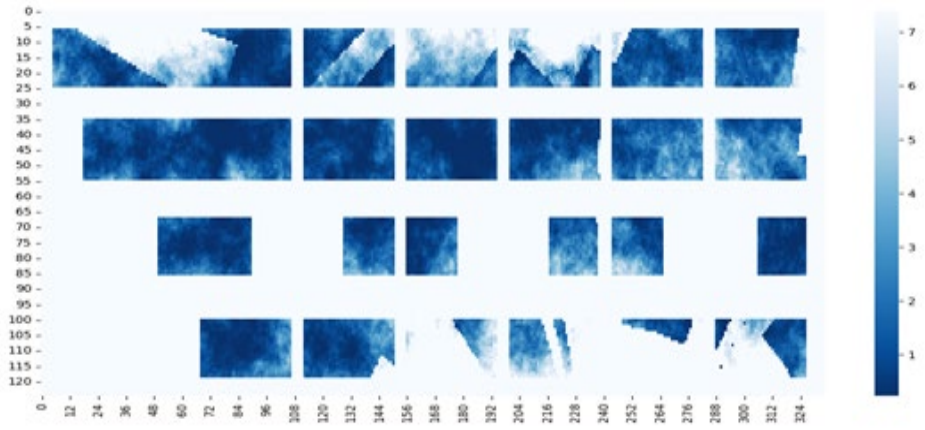


Figure 3-29: Relay-enhanced RAN control - Map of the spectral efficiency (bits/s/Hz).

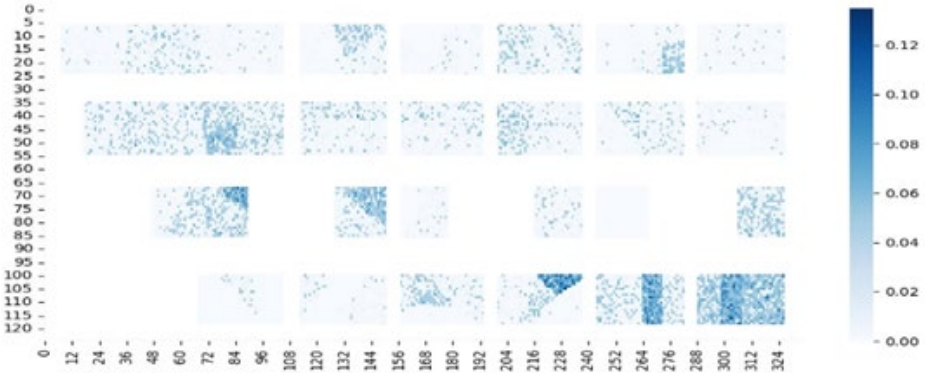


Figure 3-30: Relay-enhanced RAN control - Map of the user spatial density (UE/m2) in day 4.

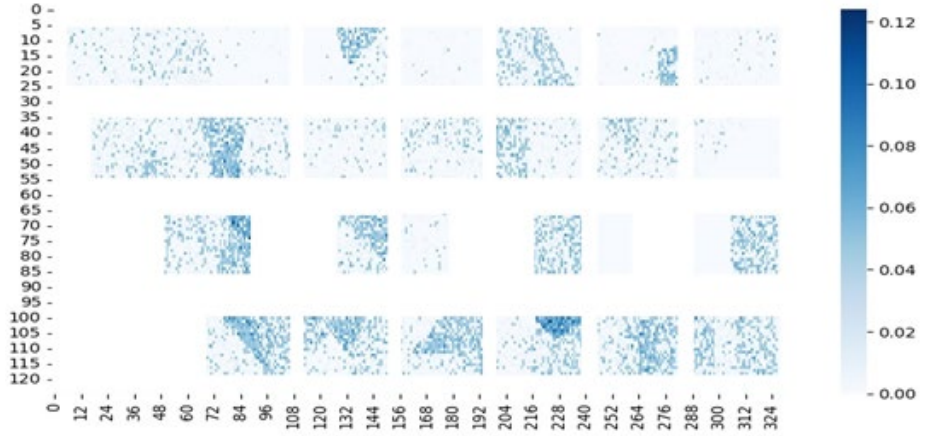


Figure 3-31: Relay-enhanced RAN control - Map of the user spatial density (UE/m2) in day 14.

Table 3-3 provides the list of the CHs identified with a repetitiveness higher than 10%. The CH repetitiveness was defined in D4.2 as the percentage of time in which the user density in the CH is higher than a specific threshold. A high CH repetitiveness indicates that a relatively large number of UEs are located inside the CH region in a large percentage of time. The CH characterization presented in Table 3-3 includes the centroid geographical coordinates, the building and the floor where each coverage hole is detected (ground floor is represented with 0), the coverage hole radius and its repetitiveness. Moreover, Figure 3-32 illustrates the geographical locations of the CHs. Note that the identified CHs correspond to geographical regions with poor spectral efficiency (see Figure 3-29) with sustained traffic levels in a relatively large number of time periods (e.g. CH_4 in B6 building). It is worth noting a very large repetitiveness in coverage holes CH_2 and CH_3 with a relatively high traffic level in more than 30% of the total observed time (which includes time periods at nights and in weekends).

Table 3-3: Relay-enhanced RAN control - Validated Coverage Holes

	Centroid Coordinates	Building (Floor)	Radius (meters)	Repetitiveness (%)
CH_1	[60,44]	C1 (floor 1)	8.6	22.91
CH_2	[77,41]	C1 (floor 0)	8	43.75
CH_3	[67,75]	B1 (floor 1)	5.5	32.29
CH_4	[318,75]	B6 (floor 0)	9.11	12.50
CH_5	[72,74]	B1 (floor 0)	7.13	15.62
CH_6	[178,42]	C3 (floor 0)	8.26	14.58
CH_7	[85,105]	A1 (floor 0)	5.4	11.45

Google. Imagery @2024 CNES/Airbus, Institut Cartogràfic de Catalunya, Maxar Technologies, Map data @2024, Spain.

Figure 3-32: Relay-enhanced RAN control - Identified CHs and fixed relay locations to address them.

3.3.2.2 Candidate RUE identification

This section explores the availability of UEs to serve as RUE to mitigate the identified CHs. For this purpose, the candidate RUE identification process described in BeGREEN D4.2 has been executed in the considered scenario.

Table 3-4 shows the percentage of time with at least one available UE to serve as a RUE for each CH. A UE is considered to be available to become RUE if the RSRP from the best serving gNB is higher than Th_{RSRP}=-90dBm. Table 3-4 also shows the average number of simultaneous UEs available to serve as RUE. These two metrics are also presented considering only the time periods when each coverage hole is detected. As shown, in CH_2 and CH_5, there is a high percentage of time with at least one available UE to serve as RUE. A relatively large number of UEs available to become RUE is also observed in these CHs. This indicates that it may be feasible to rely on RUEs to address CH_2 and CH_5. In turn, note that the RUE availability in CH_4 and CH_6 is very low, which suggests the necessity of the deployment of fixed relays for these CHs.

Table 3-4: Relay-enhanced RAN control - Statistics of UE availability to serve as RUE

	Percentage of time with at least one available UE to serve as RUE.		Average number of simultaneous available UEs to serve as RUE	
	All time periods	Only CH periods	All time periods	Only CH periods
CH_1	69.42	74.76	3.05	3.50
CH_2	85.26	85.74	4.20	4.25
CH_3	50.64	54.94	2.72	2.85
CH_4	23.95	45.83	1.74	1.79
CH_5	84.42	96.01	3.48	4.59
CH_6	11.80	22.98	1.31	1.44
CH_7	31.79	56.38	9.18	10.40

3.3.2.3 Impact of the different RUE activation policies

This section presents the UE performance when considering the different RUE activation policies. The RUE activation process is in charge of deciding dynamically which of the candidate RUEs are activated. For this purpose, the RUE activation process checks the availability of the different candidate RUEs to address the different CHs. Only candidate RUEs with an average RSRP from the serving gNB higher than a threshold Th_{RSRP}=-90dBm are considered to be available to serve as RUE. Other conditions may also be included to check the candidate RUE availability (e.g. battery level higher than a threshold, etc.). Then, the RUE activation process determines the RUE(s) that are activated to address each CH. Different policies can be considered:

1. *Activation of the RUE with the best spectral efficiency.* This policy activates the available candidate

RUE with the highest spectral efficiency from the best serving gNB. This policy aims to maximise the capacity of the gNB-RUE link.

2. *Activation of the RUE with the highest presence:* This policy activates the available candidate RUE that spends more time in the area, with the aim to minimise the number of RUE changes and, as a consequence, reduce the signalling associated to the RUE activation/deactivation. For this purpose, an active candidate RUE remains active until it becomes unavailable (e.g. until the UE moves to a different region or the UE ends its connection). In case that the active candidate RUE becomes unavailable, then the policy searches a new available candidate RUE to be activated. This is done by sorting the available candidate RUEs in a ranking according to their presence, i.e. the percentage of time in which a specific UE is present in the specified square area, defined in BeGREEN D4.2. Then, the RUE with the highest presence is activated.
3. *Activation of all the candidate RUEs that are simultaneously available*. In this case, all the available candidate RUEs are activated. This policy may improve the coverage/capacity at the CHs at expense of a higher power consumption. However, this solution would require the implementation of mechanisms to manage the possible interference among active RUEs (e.g. by coordinating the RUEs transmissions in the time domain or by using different frequency bands for each RUE).

In particular, Figure 3-33 shows the percentage of time with spectral efficiency observed by the UEs below 0.5bits/s/Hz. This metric is calculated for all the UEs located in the same building and floor of each identified CH. As shown in Figure 3-33, activating all the candidate RUEs that are available simultaneously does not lead to a clear improvement with respect to the activation of just one candidate RUE. The rationale is that the obtained coverage hole regions are relatively small when compared to the coverage area provided by a RUE. Then, the activation of just one RUE becomes enough to address the coverage hole without the need of activating additional RUEs at the same time.

Moreover, for the policies that activate just one RUE,

Table 3-5 shows the average spectral efficiency in the gNB-RUE link and the average number of different activated RUEs during the day for both policies. The policy that activates the RUE with the best spectral efficiency provides a slightly better spectral efficiency perceived by the UEs (see Figure 3-33). The rationale is that this policy allows a better spectral efficiency in the link between the gNB and the RUE (see

Table 3-5). However, an increase in the number of different active RUEs per day is observed (see

Table 3-5), which may lead to an increase in the signalling related to the RUE activation/deactivation processes.

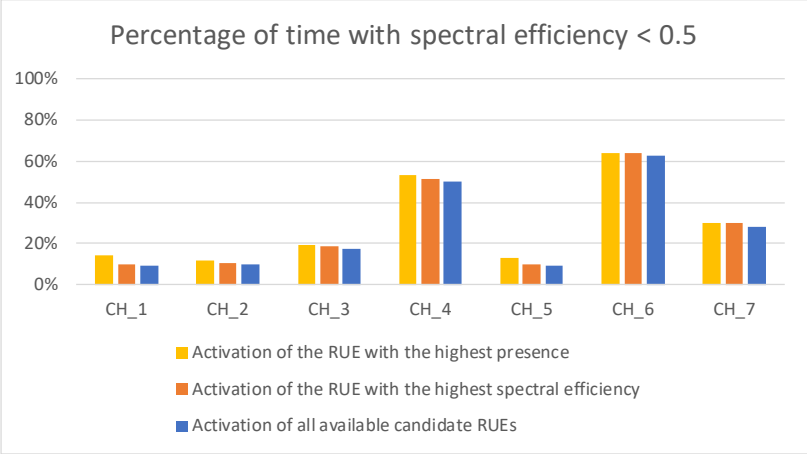


Figure 3-33: Relay-enhanced RAN control - Percentage of time with spectral efficiency observed by the UEs below 0.5bits/s/Hz in the different CHs regions.

Table 3-5: Relay-enhanced RAN control - Comparison of the RUE activation policy.

	Activation of the RUE with the highest spectral efficiency		Activation of the RUE with the highest presence	
	Avg. spectral efficiency observed by the RUE	Avg. number of different RUEs per day	Avg. spectral efficiency observed by the RUE	Avg. number of different RUEs per day
CH_1	1.96	5.95	0.96	1.80
CH_2	2.30	9.09	1.97	1.09
CH_3	1.71	6.33	1.60	3.19
CH_4	0.82	5.14	0.66	3.09
CH_5	1.67	9.09	0.80	0.38
CH_6	0.97	0.85	0.57	0.85
CH_7	3.26	11.66	2.28	5.71

3.3.2.4 Performance comparison of the different solutions

To gain insight into the potential capabilities of relays/RUEs to mitigate the CHs, this section presents a comparison of different spectral efficiency statistics observed by the UEs in the different CHs, for the benchmark case when no relays are considered, the case of deploying fixed relays and the case of using RUEs. For the case of RUEs, the RUE activation policy is the one that activates the available candidate RUE with the best spectral efficiency in the gNB-RUE link. Regarding the use of fixed relays, the relay placement algorithm described in BeGREEN D4.2 has been executed in order to identify the best location to place a fixed relay to address each CH. The obtained fixed relays locations are presented in Figure 3-32.

Focusing first on coverage hole CH_2, Figure 3-34 shows the CDF of the UE spectral efficiency observed in the geographical locations in the same building and floor where CH_2 is located (i.e. ground floor at Building C1). As shown, when no relay is deployed, the spectral efficiency is below 0.5bit/s/Hz in 40% of the UE reported measurements. The deployment of a fixed relay R_2 to address CH_2 (see Figure 3-32) clearly improves the spectral efficiency in this region (i.e. the percentage of UE measurements with spectral efficiency below 0.5bit/s/Hz is around 1.6%, as shown Figure 3-34). It is also worth noting that, for the case of deploying the fixed relay R_2, in 40% of the measurements, the spectral efficiency observed by the UEs is limited by the spectral efficiency in the link between the gNB and the fixed relay, and corresponds to 5.8 bits/s/Hz, as shown in Figure 3-34. In the case of using RUEs to address CH_2, the obtained CDF of the spectral efficiency observed by the UEs is also presented in Figure 3-34. The performance with RUEs basically depends on the availability of UEs to serve as RUE (note a high UE availability in CH_2, around 85%, as shown in Table 3-4) and the spectral efficiency in the links gNB-RUE and RUE-UE.

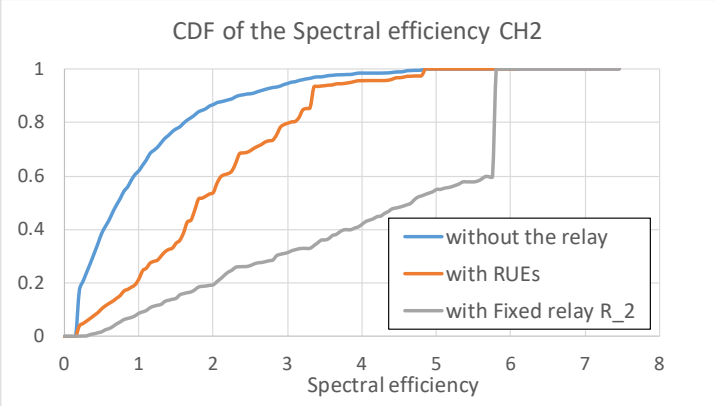


Figure 3-34: Relay-enhanced RAN control - CDF of the spectral efficiency (bits/s/Hz) observed by the UEs in the region of CH_2 (i.e. ground floor of Building C1).

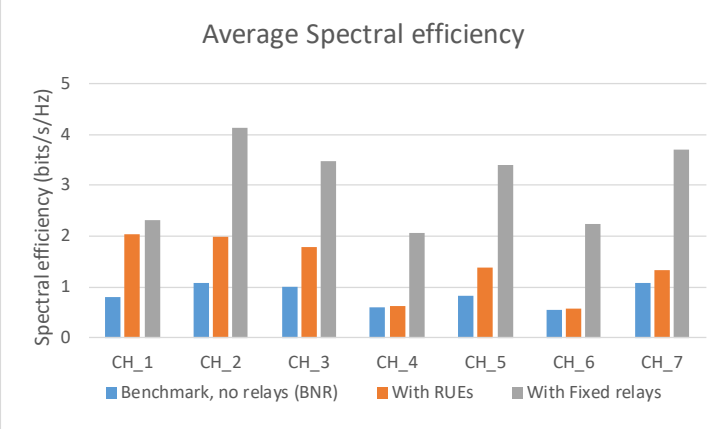


Figure 3-35: Relay-enhanced RAN control - Average UE spectral efficiency observed by the UEs.

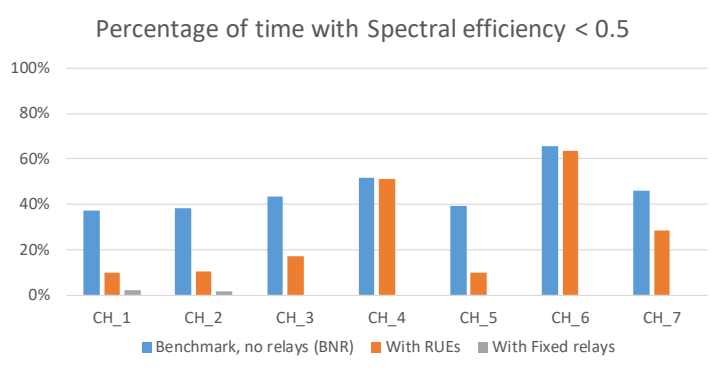


Figure 3-36: Relay-enhanced RAN control - Percentage of UE measurements with spectral efficiency below 0.5bit/s/Hz in the region of each CH.

The previous results have been extended for the different CHs regions. Figure 3-35 presents the average UE spectral efficiency and Figure 3-36 shows the percentage of UE measurements with a spectral efficiency lower than 0.5bit/s/Hz. As shown in Figure 3-35 and Figure 3-36, the deployment of fixed relays at the locations shown in Figure 3-32 clearly provides the best spectral efficiency statistics.

In some coverage holes such as CH_2 and CH_5, the use of RUEs leads to significant improvements with respect to the case of no relays (see Figure 3-35 and Figure 3-36) because of a relatively high presence of available UEs to serve as RUE (see Table 3-4) and a relatively good spectral efficiency in the links gNB-RUE and RUE-UE. The obtained performance with RUEs in CH_2 and CH_5 is not so good as in the case of fixed relays but the use of RUEs is a cost-effective solution in terms of CAPEX. It is also worth mentioning that the

performance of RUEs in CH_4 and CH_6 is very similar to the case of no relays (see Figure 3-35 and Figure 3-36) because of the low availability of UEs to serve as RUEs in these regions (see Table 3-4).

Considering a performance requirement of providing a spectral efficiency higher than Th_{Speff}=*0.5bits/s/Hz* in *P=90%* of the time, it can be observed from Figure 3-36 that the use of RUEs can satisfy this requirement in CH_1, CH_2 and CH_5. However, the deployment of fixed relays becomes necessary in CH_3, CH_4, CH_6 and CH_7 to satisfy this performance requirement. Then, the solution that combines the deployment of fixed relays and the use of RUEs (i.e. the FRR solution described in section 3.3.2), considers the mitigation of CH_1, CH_2 and CH_5 by means of RUEs and the deployment of fixed relays to address CH_3, CH_4, CH_6 and CH_7.

3.3.2.5 Power Consumption evaluation

This section aims to provide a comparison of the different CH mitigation solutions by means of relays in terms of power consumption. As shown in the previous section, better values of spectral efficiency can be obtained when using relays/RUEs, because of the more favourable propagation conditions in the involved links. This will have a relevant impact in the number of required RBs to satisfy the UE bit rate demands and, as a consequence, in the power consumption, because the transmitted power is proportional to the number of occupied RBs, see equations (A1-7) and (A1-9) in Appendix 1.

To illustrate this, first, Figure 3-37 presents the CDF of the percentage of RBs used in gNB2 for the different solutions, namely the benchmark case when no relays are considered (BNR), the solution based on fixed relays and RUEs (FRR) and the solutions based on deploying fixed relays (FRAO that keeps the fixed relays always on and FROO that switches off the fixed relays with no UEs to serve). These solutions consider the same fixed relays and/or RUEs of previous sub-section. The considered UEs required bit rate is r_b=*4Mbits/s*. As shown in Figure 3-38, the use of fixed relays (note that both FRAO and FROO give the same result) clearly reduces the amount of RBs used in gNB2 with respect to the case of no relays. The FRR solution provides a result close to the case of fixed relays. Although not presented here, it has been observed that the RBs occupancy in gNB1 and gNB3 is much lower than in gNB2, because gNB2 is the one that serves most of the users in the considered scenario.

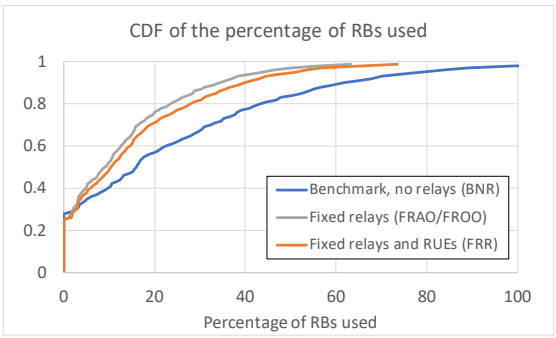


Figure 3-37 Relay-enhanced RAN control - CDF of the power saving with respect to the benchmark BNR.

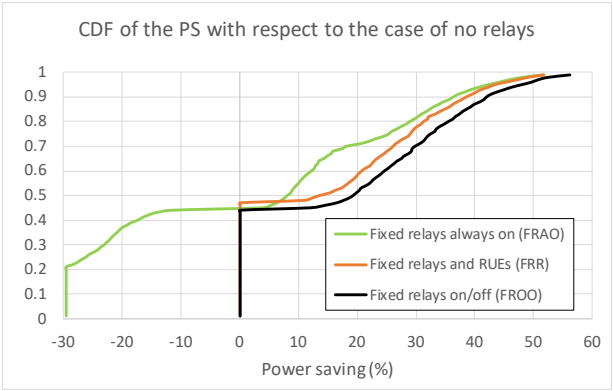


Figure 3-38 Figure 3 38: Relay-enhanced RAN control - RB occupancy at gNB2.

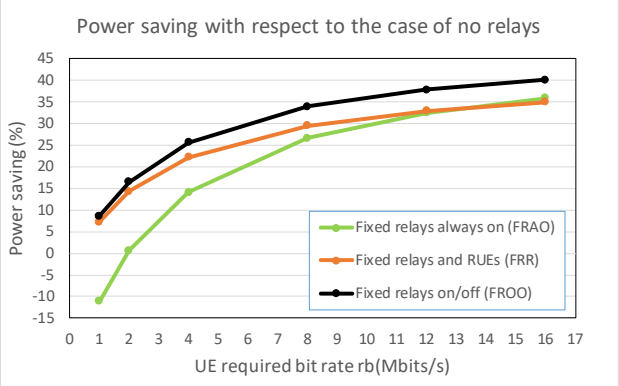


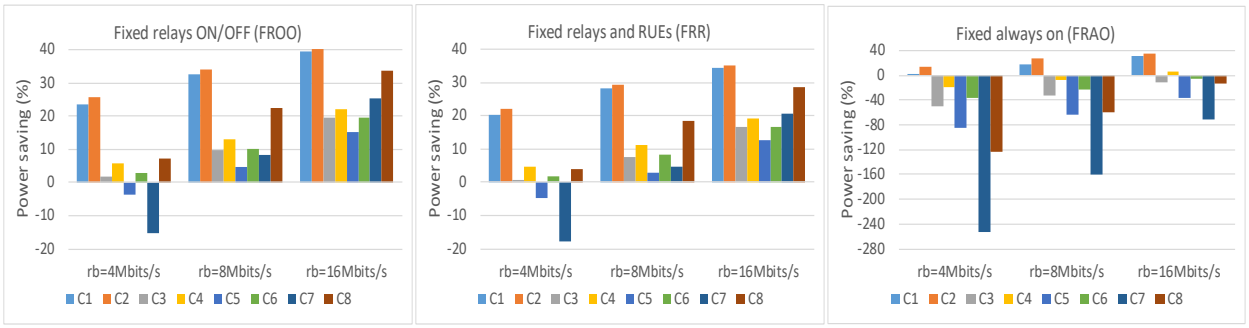
Figure 3-39: Relay-enhanced RAN control - Average value of Power saving (%) of the different solutions for different UE required bit rate.

Figure 3-38 presents the CDF of the power savings obtained with the different proposed solutions with respect to the BNR. The considered power consumption model parameters in Figure 3-38 correspond to combination C2 in Table 3-1. As shown, the FROO solution always provides a positive value of power saving with respect to the benchmark BNR. Although not shown in the Figure, it is worth mentioning that the power consumption in the benchmark case (BNR) can be up to 850W. The FROO solution can reduce this power consumption to 400W, which represents a power saving of 450W that corresponds to a reduction of 53% (see percentile 99 in Figure 3-38). The FRR solution based on fixed relays and RUEs provides power savings close to FROO.

In the case of FRAO, there is a 45% of the time in which the power saving is negative (i.e. the power consumption with FRAO is higher than BNR). This corresponds to the percentage of time in which there are no UEs served by the fixed relays. In these periods in the FRAO solution, the 7 fixed relays remain switched on (each of them consuming a power $P_{0,R}$=*6.8W*) leading to a power consumption 47.6W higher than BNR. As shown in Figure 3-38, the power saving with FRAO can be up to -30% with respect to BNR. Switching off the fixed relays with no UEs to be served (i.e. FROO solution) clearly provides benefits in terms of power savings. Focusing on average terms, the average power saving with the deployment of relays with respect to BNR ranges between a 14% for the case FRAO and a 25% for the case FROO.

Figure 3-39, illustrates the average percentage of power savings of the different solutions with respect to the benchmark BNR, for different UE required bit rates. As shown, for low values of the bit rate, the power savings are relatively low. Even negative values are obtained for the case of FRAO. The reason is that the power savings (obtained due to the better propagation conditions when deploying fixed relays with respect to BNR case) are lower than the power consumption of the relays. In turn, when the bit rate increases, the average power savings obtained with relays also increase, reaching values of 35-40% for r_b=*16Mbits/s*. The power saving of the FRR solution is very close to the case of FROO for low values of the bit rate. However, when increasing the bit rate, the lower spectral efficiencies provided by the RUEs increase the required transmitted power and the power consumption with respect to FROO, which results in lower power savings of FRR. In any case, it is worth noting relevant power savings of FRR with respect to BNR, especially when UEs demand higher bit rates.

Figure 3-40 illustrates the impact of the different combinations of the power consumption model parameters considered in Table 3-2. For this purpose, the average value of the power savings of the different relay-based solutions with respect to the BNR case of no relays have been determined for the different parameter combinations. The impact of different values of the UE required bit rate is also presented in Figure 3-40a. As shown in Figure 3-40a, the FROO solution provides positive values of power saving with respect to BNR in almost all the combinations of power consumption parameters. Higher power savings are obtained for higher values of the UEs required bit rate. In particular, as shown in Figure 3-40a, if r_b=*16Mbits/s*, the average power saving of FROO ranges between 15% for combination C5 and 40% for combination C2.



(a) (b) (c)

Figure 3-40: Relay-enhanced RAN control - Impact of the power consumption model parameters in the obtained power saving (%) with respect to the case of no relays.

The FRR solution (see Figure 3-40b) also provides promising results in terms of power savings (close to the FROO solution), and avoids the necessity of deploying fixed relays in some specific regions (namely CH_1, CH_2 and CH_5) with available UEs to serve as RUE. Finally, note also from Figure 3-40c, that the FRAO solution only provides positive power savings in combinations C1 and C2 with bit rates higher than 8Mbits/s, while negative power savings are observed in most of the combinations. This indicates the relevance of switching off the fixed relays when they are not necessary.

3.3.3 Conclusions

Previous sections have presented an evaluation of the different relay control functionalities described in BeGREEN D4.1 and D4.2, namely, the coverage hole detection, the candidate RUE identification, the fixed relay placement and the relay activation/deactivation functionalities. This evaluation has been done in terms of the UE spectral efficiency and also in terms of power consumption. The proposed algorithms have been executed in a realistic scenario that is fed with a real space/time UE geographical location distribution.

For the different identified CH regions, the UE performance achieved with a solution based on RUEs has been compared to the case of no relays and the case of deploying fixed relays. According to the obtained results, the solution based on fixed relays provides a clear improvement in terms of UE performance with respect to the case of no relays. The performance obtained with RUEs is subject to the availability of UEs to serve as RUE, and the spectral efficiency in the gNB-RUE and RUE-UE links. In some of the identified CHs, the RUE availability is rather low and, as a consequence, the performance of the RUE based solution is very close to the case having no relay. In such situations, the deployment of a fixed relay would be required. In contrast, in other CHs with higher candidate RUE availability and acceptable spectral efficiency in the gNB-RUE and RUE-UE links, the use of RUEs may allow a performance close to the case of fixed relays in a most cost-efficient way. According to this, a FRR solution that combines the use of RUEs, to address CHs with high RUE availability, and the deployment of fixed relays to mitigate the rest of the CHs has been proposed. In the considered scenario, this FRR solution guarantees a spectral efficiency higher than 0.5bits/s/Hz in more than 90% of the time for all the CHs regions, and reduces the number of fixed relays to be deployed, which leads to a CAPEX reduction.

From the point of view of power consumption, different relay-based solutions have been evaluated. It is found that the deployment of fixed relays that are switched off when no UEs are to be served (FROO solution) provides positive power savings with respect to BNR in almost all the considered situations. The power savings depend on the required bit rate and on the parameters of the power consumption model. In particular, for r_b=*16Mbits/s*, the average power saving with respect to BNR ranges between 15% and 40%, for combinations C5 and C2, respectively. In turn, when the fixed relays are always on (FRAO solution) the power savings are considerably worse, which indicates the importance of switching off the fixed relays when they are not necessary. Finally, the FRR solution that combines the use of both fixed relays and RUEs shows promising results in terms of power savings, close to the FROO solution.

3.4 Traffic-aware compute resource management to enhance UPF EE

Virtualization and softwarization of 5G network functions have accelerated the use of general-purpose hardware and commercial off-the-shelf (COTS) equipment for deploying 5G RAN and Core networks. However, this flexibility often compromises EE, particularly for traffic-intensive components like the UPF. While the Data Plane Development Kit (DPDK) offers substantial improvements in packet processing performance within UPFs, it also leads to higher energy consumption due to the heavy reliance on CPU resources.

As was presented in BeGREEN D4.2 [2], in BeGREEN we have addressed these challenges by proposing practical strategies to dynamically allocate CPU resources in DPDK-based UPFs according to traffic demands. In this section, we first complete the experimental characterization started in D4.2 regarding the trade-offs between energy consumption and performance under varying workloads, including the impact of uncore frequency scaling, which manages components such as the Last Level Cache (LLC) and Integrated Memory Controllers (IMC). These analysis leads to the proposal of different load-aware CPU allocation policies, which are applied to a scenario using real traffic statistics from a Spanish Mobile Network Operator. We then discuss the application of AI/ML-based traffic prediction to drive the application of these policies, providing a comparative analysis of different models and heuristics.

3.4.1 Experimental characterization

In BeGREEN D4.2 [2] we presented the main features and components of the evaluated DPDK-based UPF. The OAI UPF-VPP[10] is an open-source implementation of the 5G Core UPF (Release 15 & 16) based on both Vector Packet Processor (VPP) v21.01 [39] and DPDK v20.11. For clarification purposes, let us summarize below the relevant concepts and procedures impacting its performance. Figure 3-41 illustrates its high-level architecture:

- The UPF-VPP runs on top of kernel bypass technologies such as DPDK, which uses a Poll Mode Driver (PMD) that employs busy-polling to access NIC descriptors without interruptions (1 and 5 in Figure 3-41). VPP handle packets in batches, enabling accelerated packet processing in the user space (2 and 6). The vector processing nodes perform the N3/N6 GTP decapsulation/encapsulation and the forwarding to the N6/N3 interfaces (3 and 7). The output node will finally forward packets to the required NIC interface (4 and 8).
- VPP handlers packet processing and NIC polling through worker threads. Additional threads are used to scale packet processing capabilities, enabling VPP to handle higher traffic loads and match the speed of the available NICs without incurring in traffic losses.
- PMD utilization leads to full CPU usage regardless of the network load, resulting in high energy consumption and lower energy efficiency during off-peak periods. Initial results for the performance and energy consumption of the UPF-VPP under default CPU governors are shown in Figure 3-42. The powersave governor algorithm reduces CPU frequencies to the minimum configured value (1 GHz in this case), decreasing energy consumption but penalizing performance (maximum of 23.6 Gbps) due to worker capacity saturation. On the contrary, the performance governor scales CPU frequencies up to the maximum configured value (2.7 GHz in this case), maximizing performance (up to 35 Gbps) but significantly increasing energy consumption, even during idle phases without traffic. As shown in Figure 3-42b, energy consumption is also heavily influenced by CPU frequency when scaling up the number of threads.

[10] https://gitlab.eurecom.fr/oai/cn5g/oai-cn5g-upf-vpp

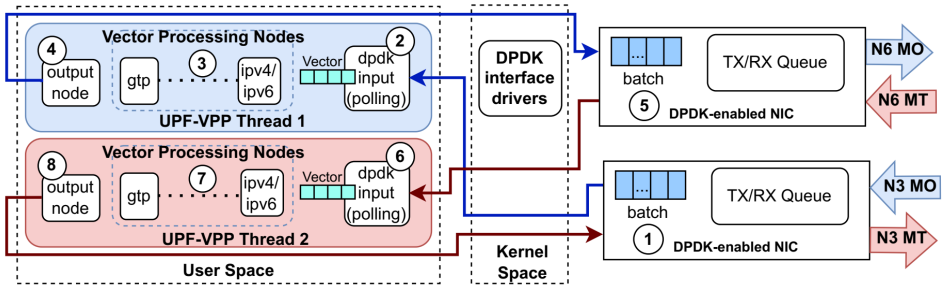


Figure 3-41: UPF-VPP - DPDK-based UPF-VPP architecture

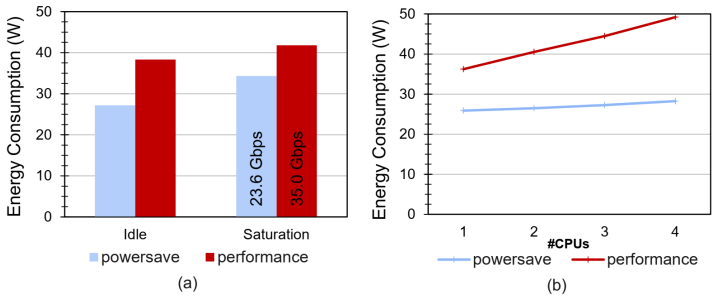


Figure 3-42: UPF-VPP - Energy consumption of the UPF-VPP with default CPU governors: (a) Single CPU in idle (without traffic) and saturation status (input traffic of ~35 Gbps), (b) Multiple CPUs in idle status

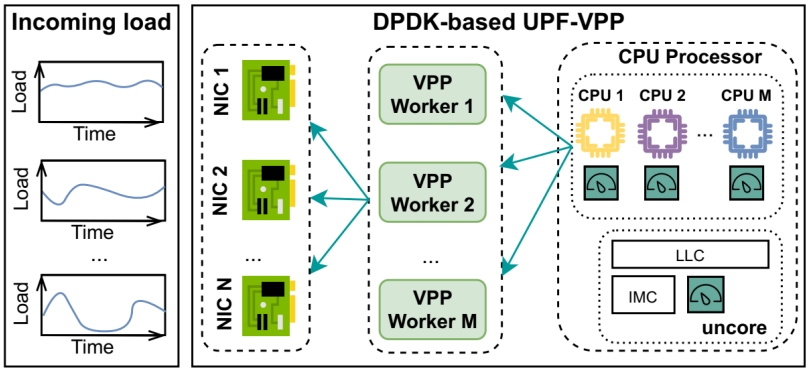


Figure 3-43: UPF-VPP - CPU resource allocation schema

The results highlight the need for energy-efficient CPU resource management according to traffic load. Three main strategies are considered, as illustrated in Figure 3-43:

- **CPU frequency scaling**: Although it is not possible to modify DPDK's PMD behaviour, which exhaustively uses CPU resources, P-states policies in the operating system can be applied to dynamically adjust CPU frequencies according to the incoming load on the NICs.
- **Uncore frequency scaling:** The built-in algorithm responsible for scaling the uncore frequency of Intel processors mainly considers the workload and the frequency of active cores [40][41]. However, increasing the uncore frequency does not always benefit packet processing on the UPF and may instead result in an energy consumption penalty. Adjusting the uncore frequency in the operating system to align with UPF requirements can help mitigate this issue.
- **NIC-Worker allocation**: The processing capacity of the DPDK-enabled NICs can be dynamically adjusted by allocating NIC's RX queues to workers, which directly impacts the utilization of worker-CPU resources. A NIC with increasing traffic demand can rely on multiple RX queues to use additional workers and CPUs. Using low-frequency cores in such scenarios could further optimize energy usage compared to relying on a single high-frequency core. Conversely, if a worker or core is not needed during a specific period, it can be reallocated to other NICs in the UPF or temporarily disabled.

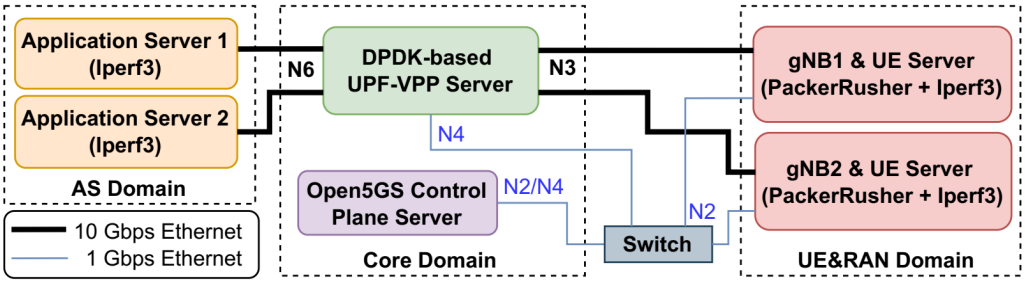


Figure 3-44: UPF-VPP - Experimental testbed setup

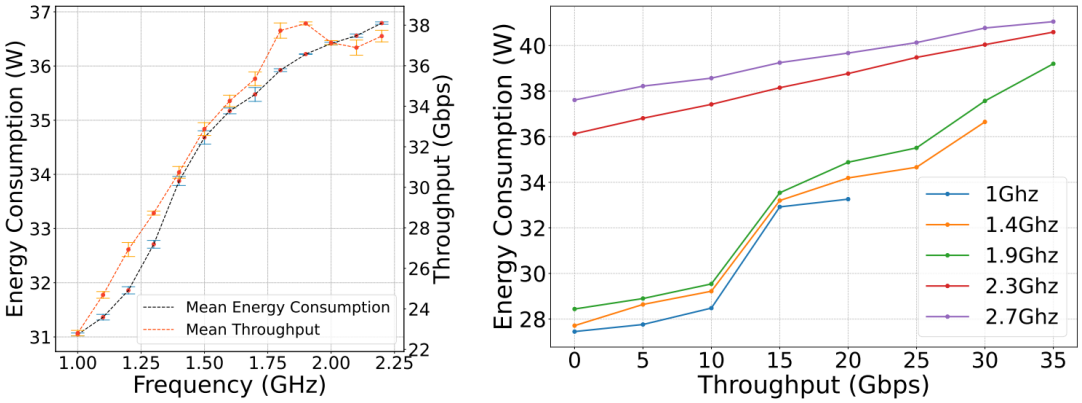


Figure 3-45: UPF-VPP - Impact of core frequency on energy consumption and throughput: (a) Saturation and (b) Throughput increase

The impact of these strategies on EE is experimentally evaluated by means of the testbed illustrated in Figure 3-44. The UPF-VPP is the Device Under Test (DUT) deployed on an X11SDV-4C-TP8F-01 server with a processor Intel Xeon(R) D-2123IT @2.2GHz (4 cores) with a Thermal Design Power (TDP) of 60W. The UE and RAN domain is based on PacketRusher[11], an open-source tool that emulates gNB and UEs, enabling high-performance testing of the control and user planes in 5G Core Networks. The Core domain includes the Open5Gs control plane and the DPDK-based UPF-VPP. Finally, the Application Server domain includes two Iperf3 servers used for generating bidirectional TCP traffic towards the UEs. The performed tests always involved balancing the traffic load among available NICs, AS, and UEs.

The energy consumption of the server hosting the UPF is measured during tests duration using powerstat[12], a Linux tool that utilizes the Running Average Power Limit (RAPL) [42] feature of Intel processors to report energy consumption across different power domains (i.e., CPU package, DRAM, etc.)

3.4.1.1 Experimental validation

First, as was reported in BeGREEN D4.2 [2], we analysed the impact of CPU frequencies on maximum or saturation throughput achievable and on the corresponding energy consumption. As shown in Figure 3-45a, the saturation throughput increases linearly with core frequency until reaching the NICs line limit of approximately 35-37 Gbps. Figure 3-45b depicts how the energy consumption increases with processed traffic and core frequency, highlighting how load-aware CPU frequency tunning can enhance energy efficiency. For instance, a core frequency of 1 GHz can process 10 Gbps of traffic using approximately 30% less energy than operating at 2.7 GHz (i.e., using the performance governor). Note however, that the improvement reduces with increasing throughput due to knee points observed at non-turbo frequencies (i.e., under 2.2 GHz).

[11] https://github.com/HewlettPackard/PacketRusher
[12] https://github.com/ColinIanKing/powerstat

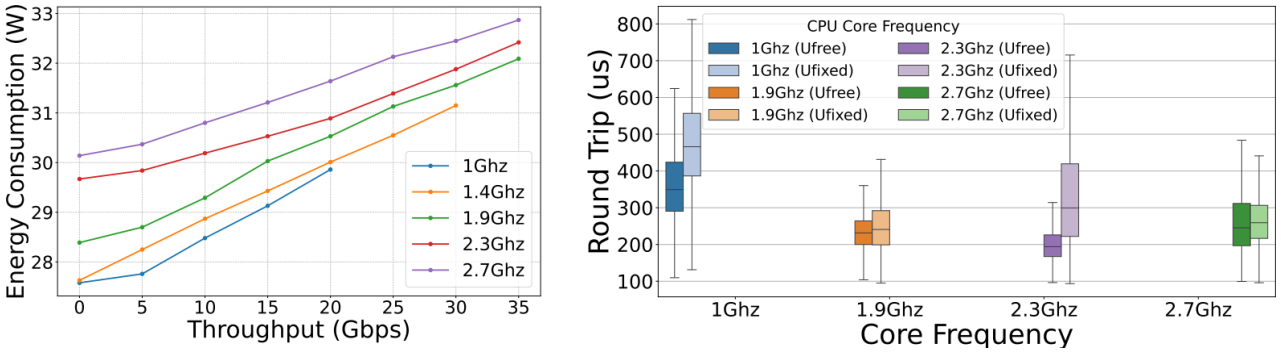


Figure 3-46: UPF-VPP - Impact of fixed uncore frequency. (a) Throughput and Energy Consumption. (b) Delay.

In our last analyses we have found that these knee points are caused by the uncore, which increases its frequency triggered by workload demands, as managed by the default uncore scaling algorithm in Intel processors. In particular, we monitored that at core frequencies below turbo levels, the uncore frequency remained at its minimum of 1.2 GHz for traffic loads under 12 Gbps. Between 12–25 Gbps, the uncore frequency scaled to 1.8 GHz, and for traffic loads above 25 Gbps, it reached its maximum value of 2 GHz. Conversely, turbo frequencies always operate at maximum uncore frequency due to PMD activity, resulting in a linear increase in energy consumption with traffic load, as shown in Figure 3-45b.

This default behaviour of the uncore leads to a significant increase in energy consumption at low loads. Therefore, we evaluated whether tuning uncore frequency can also enhance energy efficiency. Concretely, we fixed it to the minimum value of 1.2 GHz using available Linux commands (i.e., setting the *intel_uncore_frequency* file in the /sys directory) and repeated the characterization illustrated in Figure 3-45b. As show in Figure 3-46a, this configuration eliminated the knee points observed at non-turbo frequencies, resulting in a linear increase in energy consumption with throughput, while significantly reducing the energy consumption across all core frequencies (e.g., 8W in the case of turbo frequencies).

Since the uncore manages access to the LLC cache, we also analysed a possible impact on latency in the user plane due to cache misses. We measured the round-trip delay of 1E6 packets using ultraping tool[13] under a background traffic load between 20 Gbps and 25 Gbps. Obtained results, illustrated in Figure 3-46b, indicate that a fixed uncore frequency slightly increases the measured latency. Nonetheless, we can conclude that there is no significant impact on user-plane latency, as the results remain within the range of microseconds.

Once established the impact of core and uncore frequencies, we conducted additional experiments to fully characterize the contribution of different processes to the energy consumption of the UPF-VPP. First, we analysed the impact of PMD's polling operation, comparing the performance governor (Default column in Table 3-6) with cases where polling was disabled. DPDK polling was disabled through two methods: (i) Increasing the PMD's usleep period to its maximum value, which effectively disables polling and affects both core and uncore processes ("Polling Disabled (usleep)" column), and (ii) Deactivating the DPDK plugin responsible for RX queue processing, which only impacts core processes ("Polling Disabled (plugin)" column). Furthermore, two uncore frequency configurations were evaluated: (i) free mode with the performance governor (2 GHz) and (ii) fixed mode at the minimum value (1.2 GHz). The findings summarized in Table 3-6indicate the following when the performance governor is used: (i) The baseline power consumption of UPF-VPP is approximately 4 W, (ii) enabling active polling increases power usage by 4 W in idle conditions for core processes, and (iii) active polling contributes an additional power consumption of 4 W to 12 W for uncore processes, depending on the uncore frequency. It is important to highlight that the power impact described in point (ii) varies according to traffic load and core frequency, as shown in Figure 3-46.

[13] https://github.com/mrahtz/ultra_ping

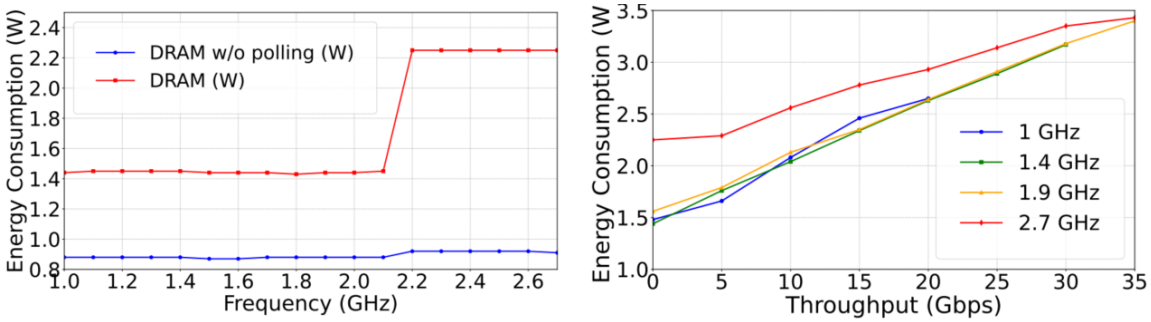


Figure 3-47: UPF-VPP - DRAM energy consumption. (a) Idle status, with and without polling. (b) Busy status.

Table 3-6: UPF-VPP - Impact of DPDK Polling on Energy Consumption Using the Performance Governor

Default	Polling Disabled (plugin)	Polling Disabled (usleep)	Baseline Server (UPF stopped)	Uncore frequency
36.11 W	32.20 W	20.28 W	16 W	2 GHz (Default)
28.64 W	24.79 W	20.19 W	16 W	1.2 GHz (Min)

We also analysed DRAM energy consumption as a function of load and CPU frequency. Figure 3-47a illustrates how DPDK polling affects DRAM energy consumption at both low and high CPU frequencies when the UPF is idle. Note that the energy penalty is significantly higher at turbo frequencies, contributing to the overall penalty of using high frequencies in worker threads observed in previous experiments. Additional tests were performed using a fixed uncore frequency, but no noticeable impact was detected. This suggests that the metric reported by *powerstat* tool is primarily associated with DRAM activity resulting from the server's workload. We also evaluated DRAM energy consumption under varying workloads, which rises linearly with increasing input load across different CPU frequencies as shown in Figure 3-47b.

To complement the analysis done in BeGREEN D4.2 [2] regarding multi-threading scenarios, we realized additional experiments fixing the uncore frequency. Therefore, we characterized UPF performance across different NIC-worker combinations under different workloads, core frequencies and uncore frequencies. Figure 3-48 presents three different configurations: (a) a single worker managing all NICs, as discussed in previous sections; (b) one worker assigned to each pair of NICs; and (c) one worker dedicated to each NIC. For simplicity, the figure assumes all cores are operating at the same frequency. However, alternative configurations have also been evaluated, which may offer slightly improved performance.

The results emphasize the effectiveness of the fixed uncore policy (U_{fix} cases), which consistently reduces energy consumption across all configurations. Additionally, comparing various threading configurations reveals that, in some scenarios, using two or four threads at low frequencies provides slightly better energy efficiency than a single thread operating at higher frequencies. For example, at a core frequency of 1 GHz, where throughput reaches 35 Gbps with two or four cores, the number of threads has minimal impact on energy consumption. This validates the multi-threading approach not only as a way to achieve higher throughputs but also to decrease energy consumption.

It is worth noting that the latest DPDK versions support dynamic RX-queue reallocation, allowing runtime RX-queue creation for already established workers. However, the UPF-VPP version used in our tests only supported reallocating pre-created RX-queues. This feature will be key for dynamically managing NIC-worker allocation based on changing load conditions.

3.4.1.2 Application to realistic traffic scenarios

Building on the findings from the previous section, we introduce a set of adaptive CPU allocation policies designed to optimize energy consumption under realistic traffic conditions. To evaluate their impact, we analysed the possible energy savings according to the traffic pattern of a dataset from a Spanish MNO. This dataset captures uplink and downlink throughput data from the Packet Data Network Gateway (P-GW) in a specific region of Spain, covering hundreds of cells over several months, with a 15-minute time granularity.

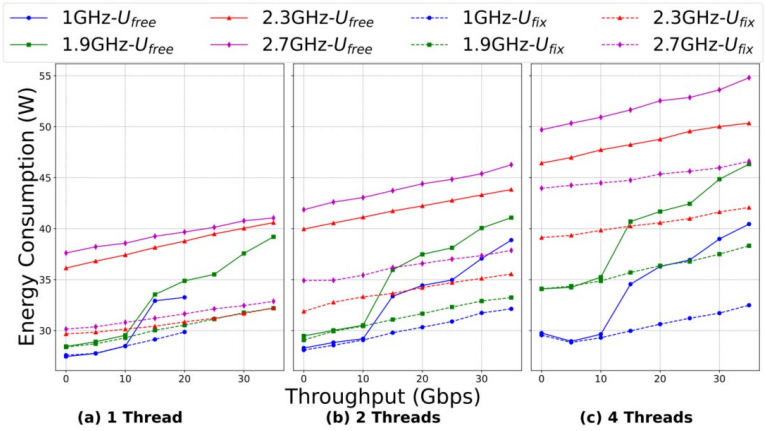


Figure 3-48: UPF-VPP - Multi-threading performance

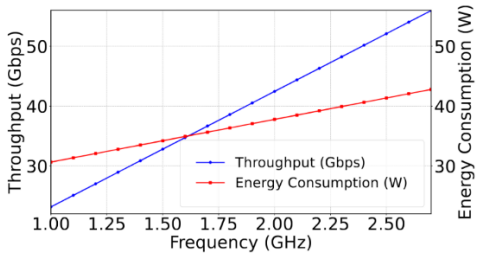


Figure 3-49: UPF-VPP - Throughput forecast and estimated energy consumption of the DUT server, assuming no NIC limitations.

Due to NIC limitations, the DUT was unable to sustain the peak throughput observed in the dataset. To address this, we estimated the necessary CPU resources to accommodate the traffic demand. Specifically, we consider the linear relationship between throughput, energy consumption and CPU frequency characterized in the previous section, which is illustrated in Figure 3-49. According to it, achieving the dataset's peak aggregated throughput of 120 Gbps would require a minimum of three cores and worker threads. We assume the incoming load is evenly distributed across all available interfaces, which, for the DUT, could be either 100 or 50 Gigabit Ethernets.

Next, we define a formula to estimate energy consumption based on the forecasted traffic and allocated CPU resources, derived empirically from measurements described in the previous sections. Given that the UPF is deployed with *N* cores and worker threads, the total energy consumption of the server hosting the UPF under a load *L* can be expressed as follows:

$$EC_{total}(L) = \sum_{i=1}^{N} EC_{UPF,workers}^{fc_i,fu_i}(L_i) + EC_{UPF,base} + EC_{server} \quad (3\text{-}6)$$

where $EC_{UPF,workers}^{fc_i,fu_i}(L_i)$ is the energy consumed by CPU *i* at core frequency fc_i and uncore frequency fu_i assigned to a worker thread in the UPF-VPP to poll and process a portion L_i of the total load *L*, $EC_{UPF,base}$ is the baseline consumption of the UPF-VPP (approximately 4 W for the DUT), and EC_{sever} is the energy consumed by the server for other operations (approximately 16 W for the DUT). $EC_{UPF,workers}^{fc_i,fu_i}(L_i)$ includes any energy consumed by the system to process the load L_i, which we found to increase linearly with the load under a fixed uncore frequency as was shown in Figure 3-46. This is consistent with state-of-the-art models, such as the linear CPU-dependent model, which predicts power consumption based on the host's CPU utilization [43]. Also, note that in multi-socket servers, the assigned CPUs could have different uncore frequencies.

The CPU allocation policies outlined in Table 3-7 are designed to enhance energy efficiency by dynamically adjusting CPU resources (i.e., CPU frequency, the number of active CPUs, and the uncore frequency) based on traffic patterns. Using these policies and the observed traffic, we applied (3-6) to calculate energy

consumption and compared the results to a non-optimized scenario where three CPUs are overprovisioned at maximum frequency to accommodate peak throughput. For simplicity, the defined policies categorize traffic into throughput ranges and assign a uniform frequency to all CPUs within each range. However, a more granular approach could be implemented to achieve further energy savings.

Table 3-7: UPF-VPP - Defined CPU Allocation Policies Based on Total Throughput

Policy	Configuration	Throughput Limits (Gbps)
P1	1 CPU at 1 GHz	< 23
P2	1 CPU at 1.8 GHz	23 ≤ x < 38.59
P3	2 CPUs at 1 GHz	38.59 ≤ x < 46.3
P4	2 CPUs at 1.4 GHz	46.3 ≤ x < 61.74
P5	3 CPUs at 1 GHz	61.74 ≤ x < 69
P6	3 CPUs at 1.9 GHz	69 ≤ x < 120

Figure 3-50 illustrates the results for a specific day, showing the energy consumption of the default policy (in orange) compared to the optimized policies (in blue). The gap between the two trends highlights the substantial energy savings achieved through traffic-aware CPU allocation. Notably, the dataset exhibited a highly regular weekly traffic pattern, leading to consistent results across different days. Based on these findings, the proposed approach can reduce energy consumption by an average of 30%, equating to approximately 11 kWh over a week for this DUT.

These results consider a prior or oracle-based knowledge of the traffic trends. In the following section we will analyse the impact of simple heuristics and of traffic forecasting on the achievable energy savings, providing a more realistic implementation of this approach.

3.4.2 AI/ML-driven traffic-aware policy management

In this section, we explore the benefits and considerations of applying two different ML models for forecasting input UPF traffic for policy management. Specifically, we consider the FB Prophet and the XGBoost frameworks. FB Prophet is an open-source time series forecasting tool based on an additive regression model[14]. On the other hand, Extreme Gradient Boosting (XGBoost) is a high-performance, distributed machine learning library designed for gradient-boosted decision trees (GBDT)[15]. XGBoost supports parallel tree boosting and is widely used for regression, classification, and ranking tasks due to its scalability and efficiency. The following section provides a more detailed description and validation of each model.

3.4.2.1 Model validation

FB Prophet was developed by Facebook for time series forecasting. The time series, *y(t)*, can be decomposed into multiple components as follows:

$$y(t) = g(t) + s(t) + h(y) + \varepsilon_t$$

In this formulation, *g(t)* represents the trend function, capturing non-periodic variations in the time series. The term *s(t)* accounts for periodic fluctuations, such as weekly or annual seasonality. The function *h(t)* models the influence of holidays, which may occur irregularly over varying durations.

[14] https://facebook.github.io/prophet/

[15] https://xgboost.readthedocs.io/en/stable/python/python_intro.html

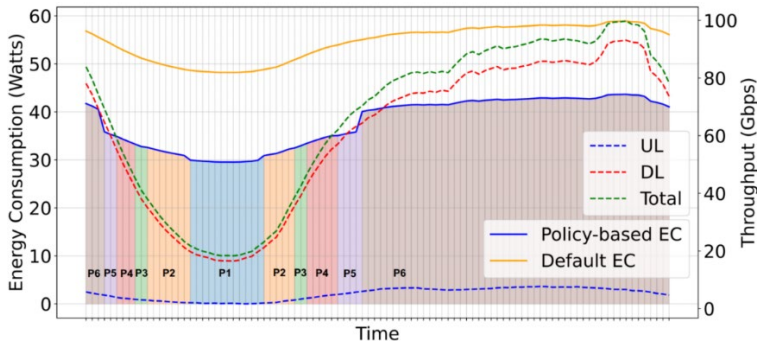


Figure 3-50: UPF-VPP - Application of CPU resource allocation strategies to a realistic traffic scenario.

The error term ε_t encapsulates any stochastic variations not explicitly captured by the model [44]. Prophet allows forecasting the UPF load for a specific time period based on historical data and identified trends, without requiring additional input features.

XGBoost can also be used for time series forecasting, which seems a useful approach due to the clear correlation of load with the time found in our dataset, as was introduced in BeGREEN D4.1 [1]. In the case of XGBoost, the traffic forecasting model uses lagged values from previous intervals as features. In our case, we used the most recent value as input for the prediction (i.e., the value of the last 15 minutes interval).

These differences between both models affect their performance in the UPF use case, as analysed below. For both models, the training period spans one full month, and the testing period also lasts for one month. As shown in Figure 3-47 and Figure 3-51, both models show similar during a regular week without anomalies; however, FB Prophet performs worse during an anomalous week in the dataset (referred to as the 'anomalous week' in this section). This performance drop is due to FB Prophet's lacking the inclusion of lagged load values as input features. Figure 3-52 and Figure 3-53 illustrate in detail the performance during this week.

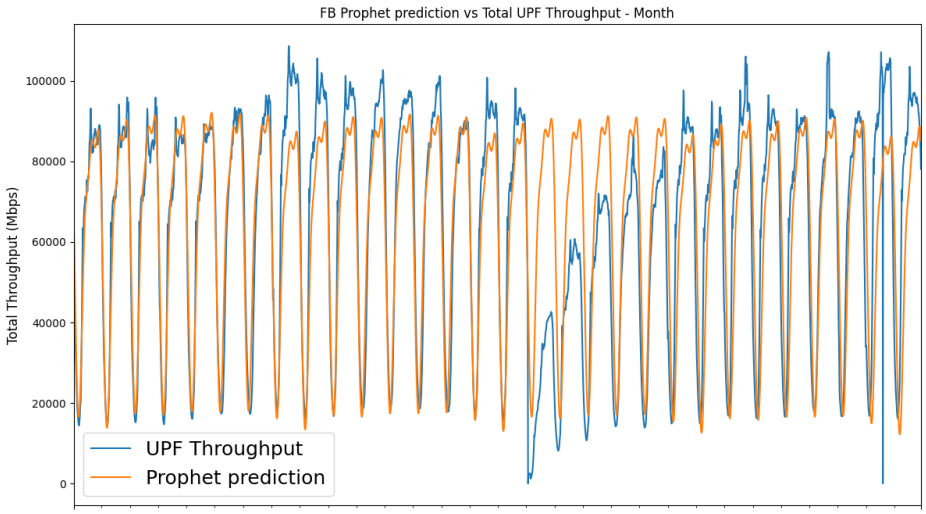


Figure 3-47: UPF VPP - FB Prophet prediction vs Total UPF Throughput (month)

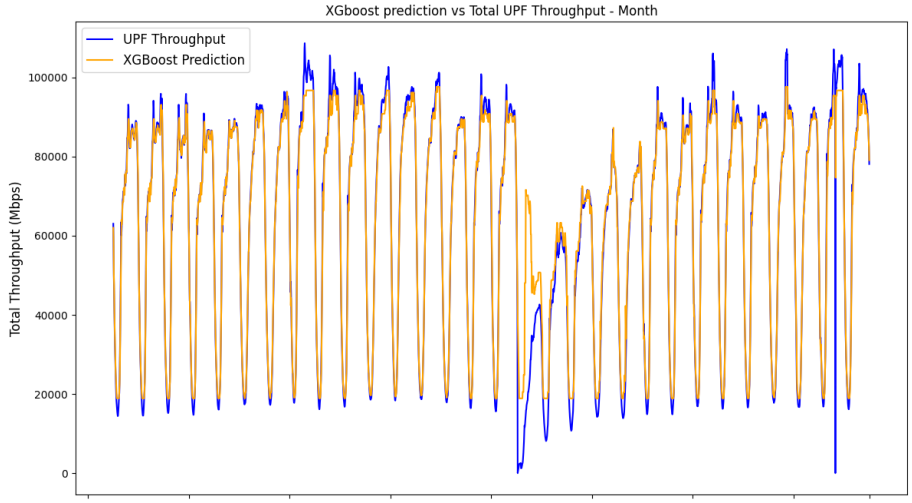


Figure 3-51: UPF VPP - XGBoost prediction vs Total UPF Throughput (month)

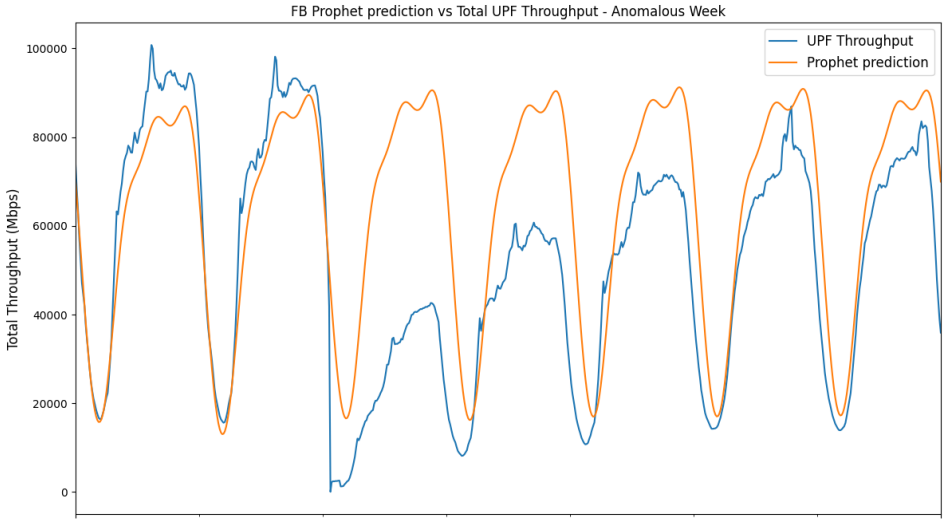


Figure 3-52: UPF VPP - FB Prophet prediction vs Total UPF Throughput (Anomalous week)

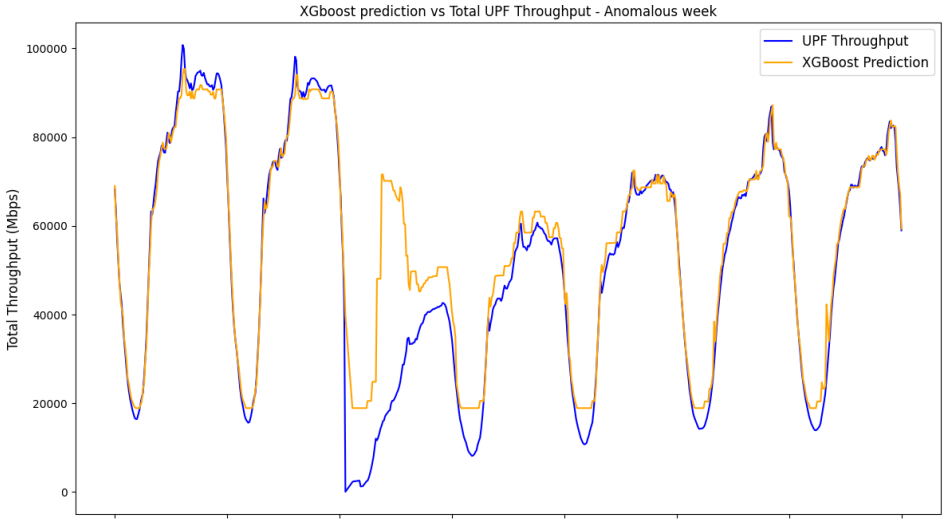


Figure 3-53: UPF VPP - XGBoost prediction vs Total UPF Throughput (Anomalous week)

Table 3-8 shows the MAE and R^2 value of both models during the whole month, an average week and the anomalous week. The MAE results for Prophet show an approximate 8% error (around 10 Gbps) over peak values. While Prophet is easy to use and deploy, its accuracy tends to degrade when predictions extend

further from the training window. In our case, Prophet's trend component remains relatively flat over time due to training on a one-month historical period. This makes it suitable for stable patterns with a consistent trend but leads to significant errors during periods of abnormal and non-periodic traffic, as is the case of the 'anomalous week. To improve accuracy in such scenarios, the model would need to be re-fitted, requiring additional computational resources. In contrast, the XGBoost model, achieves a lower MAE of around 2% (approximately 2.5 Gbps) over peak values. It maintains stronger accuracy on future testing data without needing retraining, as it consistently incorporates recent lagged values that capture the latest traffic dynamics. This makes it less prone to performance degradation. For the anomalous week, XGBoost performance could potentially be improved further by including additional features, such as the number of active data users in the UPF or increasing lag granularity.

Table 3-8: UPF-VPP – Comparison of FB Prophet and XGBoost Models

Model	MAE (Mbps)	R^2
FB Prophet: 1 Month	10669.38	0.717
XGBoost: 1 month	2555.67	0.96
FB Prophet: Average Week	8063.28	0.879
XGBoost: Average Week	1952.19	0.9914
FB Prophet: Anomalous week	16525.44	0.33147
XGBoost: Anomalous Week	4937.25	0.853

3.4.2.2 Model evaluation in the use case

In this section we analyse the impact of the traffic forecasting on the CPU management policies presented in Section 3.4.1.2. For each model, we computed the saved energy for an average week in the testing month and for the anomalous week when using the forecasted load to decide on the applied policy. As shown in Table 3-9, the percentage of saved energy is around the 30% when applying both models, being very close to the oracle case considered in Section 3.4.1.2. As expected according to previous analysis, FB Prophet results are worst during the anomalous week.

Table 3-9: UPF VPP – Impact of Model Forecasting on Saved Energy When Applying CPU Management Policies

Model	Energy Saved (KWh/week)	Energy Saved (%)
Average Week		
Oracle	11.08	30.39
FB Prophet	10.93	29.97
XGBoost	11.05	30.30
Anomalous Week		
Oracle	11.59	33.15
FB Prophet	9.56	27.34
XGBoost	11.31	32.34

However, beyond the energy-saving analysis, it is also important to consider the impact on served throughput when restrictive policies are applied in this scenario. To this end, we analysed three performance metrics:

- Correct Policy Allocation: The policy according to forecasted traffic matches the oracle policy, which is based on the actual observed throughput.
- OverSLA Policy Allocation: The policy according to forecasted traffic is more conservative than the oracle policy due to an overestimation of future traffic. This results in higher energy consumption

compared to the oracle. In this case, we compute the additional energy consumption (EC wasted).

- UnderSLA Policy Allocation: The forecasted policy is more aggressive than the oracle policy due to an underestimation of future traffic. This leads to under-served throughput, which is also computed.

Table 3-10 shows the results of this analysis for the average and anomalous weeks, showing how XGBoost outperforms FB Prophet in both cases and in all metrics. Note that the OverSLA decisions usually lead to a low increase of energy wasted. As discussed in Section 3.4.1.2, applying any CPU management policy results in significant baseline energy savings compared to the default use case, while the differences between individual policies are less pronounced for the observed throughput levels in the dataset. Therefore, erroneous policies do not lead to significant levels of wasted energy except for FB Prophet during the anomalous week due to its high throughput overestimation, as shown in Figure 3-52. The impact on throughput due to underestimations is also more pronounced in the case of FB Prophet, as it fails to accurately capture traffic peaks. On the contrary, the performance of XGBoost is very good even in during the anomalous week.

Table 3-10: UPF VPP – Impact of Model Forecasting on CPU Management Policies Performance

Model	Correct	OverSLA	Energy Wasted	UnderSLA	Under-Served Throughput
Average week					
FB Prophet	76.19 %	14.70 %	0.94 %	9.0 %	8.57 %
XGBoost	95.00 %	3.20 %	0.20 %	1.6 %	1.30 %
Anomalous week					
FB Prophet	44.9 %	50.5 %	8.42 %	4.1 %	5.30 %
XGBoost	86.0 %	13.3 %	0.77 %	0.6 %	0.78 %

Figure 3-54 shows how the policies driven by the different models impact the energy consumption and the served throughput of the UPF during a selected day in the dataset. As previously discussed, the policies based on XGBoost predictions closely track the actual traffic trends. In contrast, FB Prophet tends to underestimate throughput peaks, resulting in reduced energy consumption but at the expense of under-served throughput.

To further reduce UnderSLA errors, we considered applying the MAE obtained during the training phase as a correction delta added to the forecasted traffic. As shown in Table 3-11, the number of correct decisions decreases in both cases due to an increase in OverSLA allocations. However, the resulting impact on energy waste remains minimal, due to the baseline energy savings discussed earlier. On the other hand, UnderSLA decisions are reduced to below 1% in both cases, which is particularly beneficial for FB Prophet given its tendency to underestimate traffic peaks.

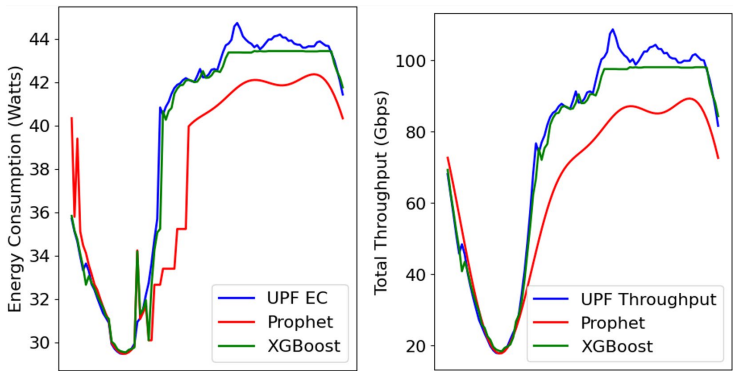


Figure 3-54: UPF VPP - Impact of model forecasting on saved energy and throughput when applying CPU management policies

Table 3-11: UPF VPP – Impact of Model Forecasting on CPU Management Policies Performance

Model	Correct	OverSLA	Energy Wasted	UnderSLA	Under-served throughput
Average week					
FB Prophet	64.8 %	34.2 %	2.90 %	0.89 %	0.80 %
XGBoost	89.2 %	10.5 %	0.63 %	0.14 %	0.14 %
Anomalous week					
FB Prophet	33.5 %	65.5 %	11.9 %	0.90 %	1.11 %
XGBoost	71.2 %	28.4 %	1.9 %	0.45 %	0.58 %

3.4.3 Conclusions

The dynamic adaptation of CPU resources to workload requirements is a key strategy for enhancing the energy efficiency of edge servers hosting network functions, such as the UPF. Throughout the various deliverables of this Work Package, we have characterized the energy consumption and performance of open-source UPFs, specifically Open5G and OAI UPF-VPP, when deployed on bare-metal COTS servers under varying traffic workloads and CPU resource allocations.

The results demonstrate that dynamic management of CPU and uncore frequencies can yield substantial energy savings without compromising traffic performance, particularly in DPDK-based user planes due to their consistently high CPU utilization. In this deliverable, we finalized the characterization of these strategies and explored the application of AI/ML-driven CPU management policies, applying traffic forecasts derived from a real dataset provided by an MNO. The results highlight the potential of these strategies, achieving approximately 30% energy savings without impacting the served throughput.

Future work should explore additional techniques, such as leveraging C-states, and the integration of more advanced AI/ML algorithms, including reinforcement learning.

3.5 Joint orchestration of vRANs and Edge AI services

In this final deliverable of WP4, we evaluate the algorithm proposed in the previous deliverable. We call our algorithm EdgeBOL. As explained in D4.2, EdgeBOL is an online learning algorithm that solves the contextual bandit problem defined in D4.2. EdgeBOL identifies a policy that minimizes the aggregate energy costs while adhering to predetermined performance criteria of the Edge AI services. Our prototype consists of a vBS, a user equipment (UE), a digital power meter, and an edge server, as shown in Figure 3-55. The vBS and UE include an NI USRP B210 as radio unit (RU) and a general-purpose computer (Intel NUCs with CPU i7-8559U@2.70GHz) deploying the near-RT RIC (for the vBS) and the baseband unit (BBU), implemented with the srsRAN suite (which emulates an O-eNB for experimentation). The vBS and UE are connected through SMA cables with 20 dB attenuators, and we adjust the transmission gain of the RU's RF chains to attain different uplink SNR values. Without loss of generality, we set 20 MHz bandwidth for the LTE interface.

Our edge server is equipped with a CPU Intel i7-8700K @ 3.70GHz and a GPU Nvidia GeForce RTX 2080 Ti. The vBS and server are connected using a switch with Gigabit Ethernet technology. To measure the power consumption at the BBU and the server, we use the digital power meter GW-Instek GPM-8213 with the GWInstek Measuring adapter GPM-001. The evaluated AI service is implemented using Detectron2[16], developed by Facebook, which performs object recognition.

[16] https://github.com/facebookresearch/detectron2

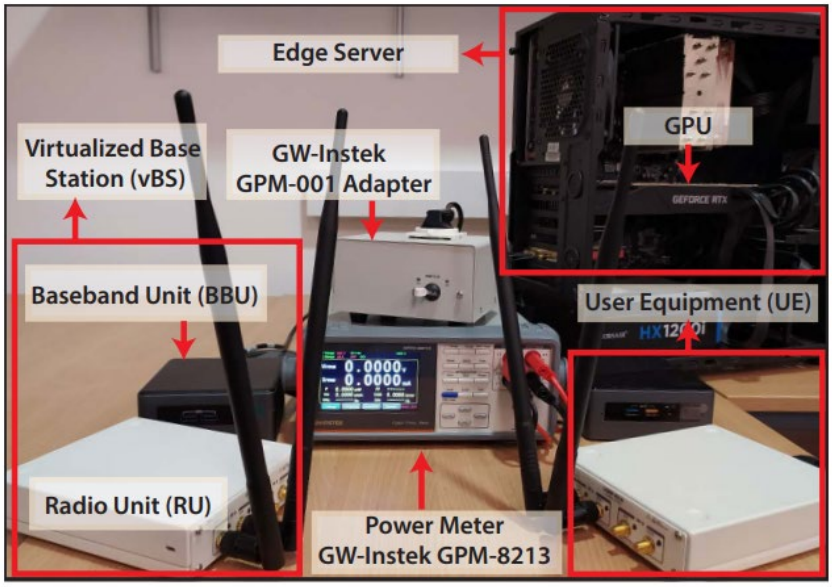


Figure 3-55: vRAN and Edge AI Services – Experimental Testbed

Specifically, Detectron2 is configured with a region-based convolutional NN (Faster R-CNN) comprising a ResNet backbone with conv4 layers and a conv5 head with a total of 101 layers. The UE sends to server images from the COCO data set we used in previous deliverables through the LTE uplink. The images are resized at the user side using the OpenCV library in Python. The bounding boxes and object classes are computed by Detectron2 and sent back to the UEs (LTE downlink).

We introduced two key srseNB modifications. First, we modified the radio MAC scheduler to implement the two radio policies we presented in D4.2. Secondly, we integrated the O-RAN E2 interface as defined in O-RAN specifications WG3 to enforce the radio control policies (MCS and airtime) on-the-fly and send consumed power consumption samples to the corresponding xApp. For the latter, we have added code into srsRAN to collect this information from the power meter. We have also implemented a proof-of-concept Near-RT RIC and Non-RT RIC with the interfaces mentioned in D4.2. We configure the GPU speed by using the Nvidia driver that allows us to set the maximum power management limit, ranging between 100 and 280W. This runtime configuration does not affect the GPU operation. Note that the actual GPU consumed power depends on its duty cycle.

We consider number of different actions ($|\mathcal{H}| = |\mathcal{A}| = |\Gamma| = |\mathcal{M}| = 11$, i.e. $\mathcal{H}$ denote the set of possible image resolutions; $\mathcal{A}$ the set of possible airtime configurations, Γ the possible GPU speed configurations; and $\mathcal{M}$ the set of all possible MCS policies as defined above); hence there is a large number of $|\mathcal{X}| = 11^4 \cong 14.6 \cdot 10^3$ control policies, which, in combination with the effect of the possible contexts, highlights the need for a data-efficient learning mechanism. Given the complexity of running experiments with multiple users, we rely on a single user in most of our experiments (which render trivial low-layer controllers). However, whenever needed (we test out multiple heterogeneous users later), we adopt simple controllers (e.g. MAC layer scheduling) that are detailed where relevant. In line with previous works [45][46], we select $\beta^{1/2} = 2.5$, which shows good performance in our evaluations. Finally, unless otherwise stated, we will plot our results with lines and shadowed areas representing, respectively, the median value and the 10th and 90th percentiles, across 10 independent repetitions.

3.5.1 Convergence evaluation

To evaluate the convergence of our approach, we consider a single context and a certain constraint set with $\rho^{\min}$ (minimum mAP performance) and $d^{\max}$ (maximum service delay). Dynamic context changes and different constraints are evaluated later.

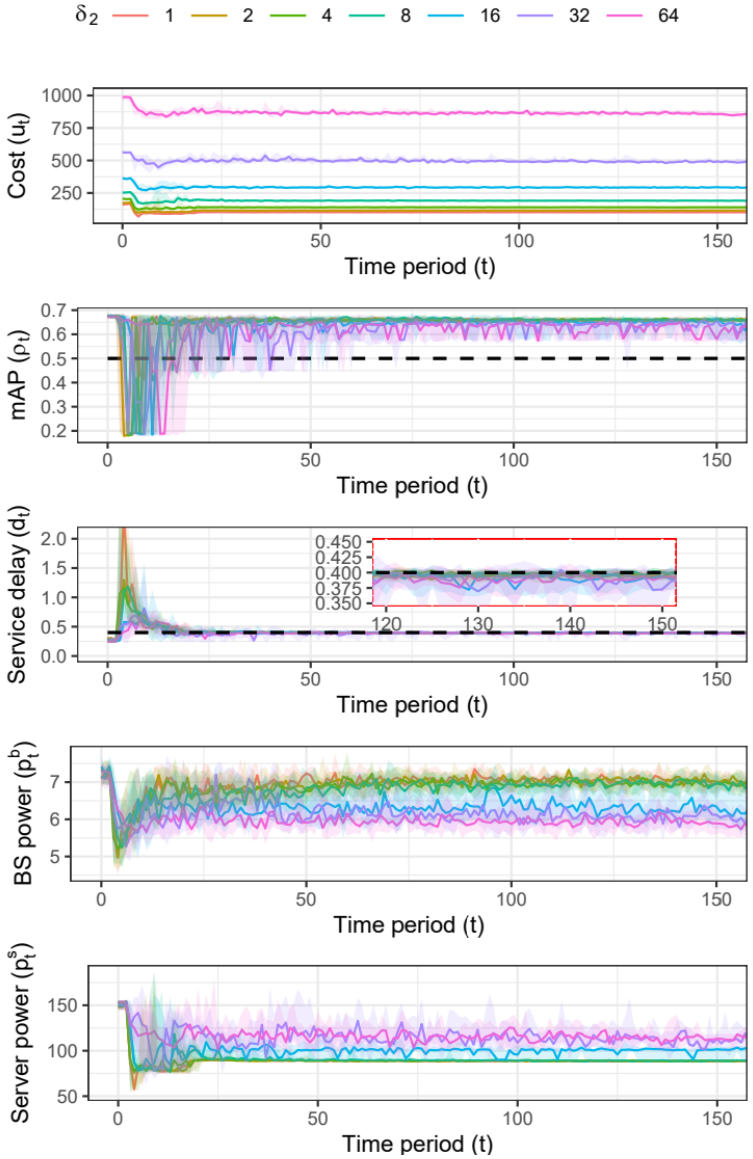


Figure 3-56: vRAN and Edge AI services - Convergence evaluation. A scenario with steady channel conditions (no context changes), δ_1=1 mu/W, $\rho^{\min} = 0.5$, and $d^{\max} = 0.4$ s.

Namely, we set the mean SNR to 35 dB (good wireless conditions), $\delta_1 = 1$ mu/W, $\rho^{\min} = 0.5$, and $d^{\max} = 0.4$ s. Figure 3-56: plots the evolution over time of the cost (u_t), mAP performance (ρ_t), delay (d_t), and server and BS power consumption (p_t^s and p_t^b) as a function of δ_2.

The first observation is that the cost u_t (top plot) converges within roughly 25 time periods across all $\delta_2 = \{1,2,4,8,16,32,64\}$. Higher δ_2 values induce higher cost as the price associated to each watt consumed by the BS grows. Remarkably, both the mAP performance and delay fall within the selected system constraints upon convergence with *high probability*. In fact, we have observed consistent results (converge speed, satisfaction of system constraints) irrespectively of the context and system constraints. The system power consumption presents interesting trade-offs with δ_2. In particular, small δ_2 (e.g., $\delta_2 = 1$) induce high power consumption at the BS but low at the server. This is because the maximum net power consumption of our virtualized BS (around 7.25 W) is much smaller to that of the server (between 85 to 180 W). Therefore, if the associated cost (mu/W) is similar for both BS and server, our solution will minimize the power consumption of the server at the expense of a small energy toll for the BS. However, when δ_2 is relatively high (e.g., $\delta_2 = 64$), the actual cost associated with the energy footprint of the BS becomes comparable, or even higher, than that of the server.

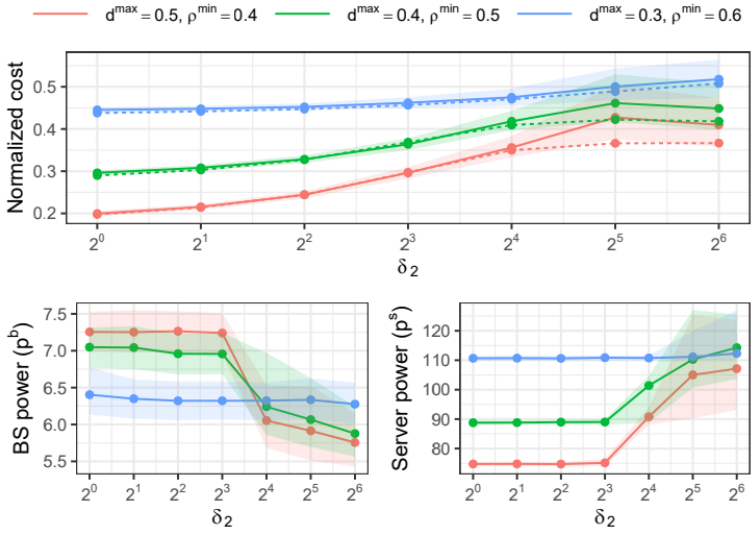


Figure 3-57: vRAN and Edge AI services - Power consumption and normalized cost for a single context as a function of δ_2, with δ_1=1 mu/W. Dashed lines represent our exhaustive search approach.

Hence, our algorithm drives the system to configurations that minimize BS power consumption at the expense of the server energy. This latter case is relevant for situations when a small cell has a stringent power budget (e.g., solar-powered) or has cooling restrictions. Indeed, different types of BS have different energy footprint, which motivates the need for approaches that learn the relationship between power consumption patterns, performance metrics and configuration policies.

3.5.2 Static scenarios

Let us now take a closer look at the power consumption and the respective policies for different constraints and values of δ_2. Figure 3-57 shows the power consumption and normalized cost once our solution has converged for $\delta_1 = 1$ and $\delta_2 = \{1,2,4,8,32\}$ mu/W. We compute the normalized cost independently for each δ_2 so we can compare across different δ_2 values. We now test different constraint settings: (i) $\rho^{\min} = 0.4$, and $d^{\max} = 0.5$ s (lax constraints), (ii) $\rho^{\min} = 0.5$, and $d^{\max} = 0.4$ s (medium constraints), and (iii) $\rho^{\min} = 0.6$, and $d^{\max} = 0.3$ s (stringent constraints), represented in red, green, and blue in the figure. In addition, we represent with dashed lines the cost attainable by an offline oracle, which we obtained using a time-consuming exhaustive search procedure over the whole control space. Though this approach is unfeasible in practice, it is a good benchmark to empirically assess the optimality of our algorithm.

Ignoring, for now, the differences across different constraints settings (different colours in the plots), we can make two observations. First, we can confirm our earlier observation that higher values of δ_2 (compared to δ_1) steer our solution to shift power consumption from the server to the BS (and *vice versa*). Second, our algorithm is able to drive the system to near-optimal points of operation, when comparing the cost of EdgeBOL with that obtained by our oracle.

In more detail, the figure renders very different behaviour across different constraint settings (colours in the plot). In the case of $\rho^{\min} = 0.4$, and $d^{\max} = 0.5$ s (lax constraints, in red in the plot), there is a drastic change in the selected policies and resulting power consumption as we increase δ_2. Because these settings are rather lax, our solution has more leeway to explore (and then select a policy from) a larger space of feasible policies. This is made evident when we compare its normalized cost with that of the most stringent settings ($d^{\max} = 0.3$ s, $\rho^{\min} = 0.6$, blue line in the figure): for $\delta_2 = 1$, the minimum cost attained by our solution is 25% larger for the latter, and 10% for $\delta_2 = 64$. Moreover, the normalized cost consistently grows, though with a shrinking gap in cost across constraint settings, as we increase δ_2. This occurs because, in our testbed, the range of power values that the BS can consume (across all policies) goes between 4 and 8 W, which is substantially smaller to that of the server (between 50 and 200 W).

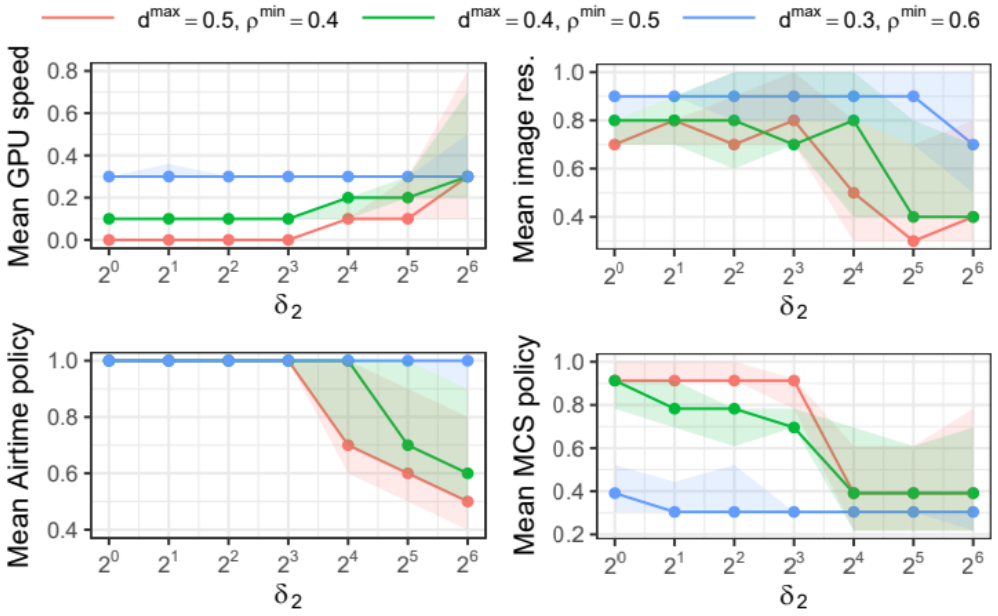


Figure 3-58: vRAN and Edge AI services - Policies for a single context as a function of δ_2, with δ_1=1 mu/W.

As a result, when we increase δ_2, i.e., when we increase the importance given to reducing BS power, the cost variance across policies reduces. This may be different with different types of BS such as macro cells.

Finally, Figure 3-58 shows the corresponding control policies for the same scenarios shown Figure 3-57. Let us first take a look to the lax settings ($d^{\max} = 0.5$ s, $\rho^{\min} = 0.4$ depicted with red lines in the figure).

When δ_2 is small, EdgeBOL imposes low-consuming server-side policies, i.e., low GPU speed policies. This certainly helps to reduce the server consumption, and the overall cost as a consequence. However, to meet the performance constraints, our solution has to compensate low GPU speed policies with higher image resolutions and higher radio policies that ease the job of the service while minimizing delay, which comes at at the expense of higher BS power consumption. Conversely, when δ_2 increases, our approach selects low-consuming radio policies and, to compensate, lower image resolutions and higher GPU speed policies that help reduce service delay. On the other side, for the scenario with most stringent constraints ($d^{\max} = 0.3$ s, $\rho^{\min} = 0.6$, blue lines), our solution is forced to deal with a smaller space of feasible policies. Therefore, all policies are roughly consistent across different δ_2 values (with mild differences for the highest settings).

3.5.3 Heterogeneous users

In an effort to reduce the problem of the dimensionality, we aggregate statistics of individual users (mean SNR, variance SNR, etc.) when describing the context. To validate that this design choice does not compromise optimality, we have performed a series of experiments with multiple heterogeneous users. Without loss of generality, we adopt simple low-level control mechanism to enforce the selected policies when allocated resources to individual users: (i) a round-robin radio scheduling approach at the MAC layer of the BS, (ii) equal image resolution across users, (iii) MCS selection approach legacy of srsRAN [47] (upper bounded by the policy), and (iv) highest GPU speed to handle individual video frames allowed by the policy.

We train the algorithm with a variable number of heterogeneous users with changing channel quality. Once trained, we evaluate the performance of our solution in scenarios with a fixed number N of heterogeneous users. The first user has the best channel conditions (SNR = 30 dB in average) and every additional user has 20% lower SNR. We trivially choose $d^{\max} = 2$ and $\rho^{\min} = 0.6$ so the system has a feasible solution in the worst case (with 6 users). Figure 3-59 depicts the cost of the system (as defined in eq. (1)) for scenarios with different values of N. We do this for different weights δ_2 in the trade-off between the service's power consumption and that of the vBS ($\delta_1 = 1$ in all scenarios).

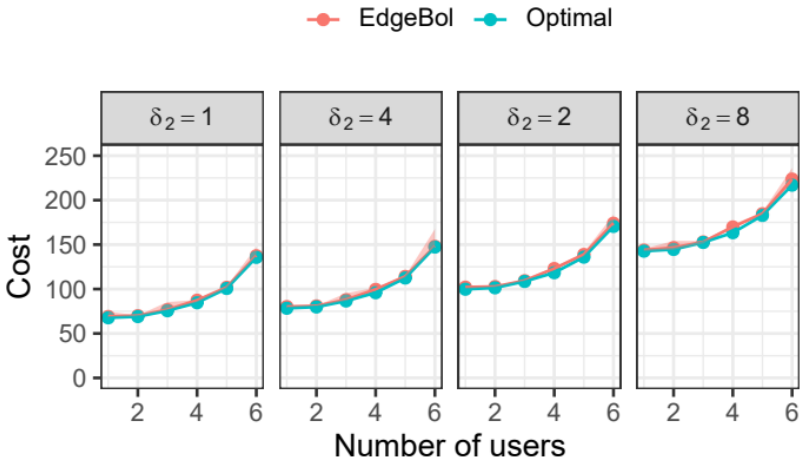


Figure 3-59: vRAN and Edge AI services - Empirical optimality gap in scenarios with multiple heterogeneous users. Each scenario has N users with different SNR conditions: user 1 has the best channel conditions (SNR = 30 dB in average) and every additional user has

We compare the performance of EdgeBOL with that of an optimal oracle that finds the best possible combination of policies offline after an exhaustive search where all the system dynamics are known. Hence, though it is unfeasible to use in practice, it provides a lower bound cost that helps us assess the optimality gap of EdgeBOL empirically.

The results show that the performance attained by EdgeBOL is remarkably close to that of the oracle, well within 2%. Though it is not shown in the plot due to space constraints, EdgeBOL satisfies the service constraints with probability 0.98. This validates that aggregated statistics across users suffice to provide good performance yet keeping the problem's complexity tractable. We can also observe that the overall cost increases with the number of users. The reason is that, as each additional user has lower SNR, its transmission time is higher. As a consequence, EdgeBOL is forced to invest more resources (i.e., airtime, GPU speed) in the system to compensate this degradation of mean wireless conditions.

3.5.4 Dynamic scenarios

We now test the performance of EdgeBOL in the presence of fast context dynamics and sudden constraint changes. To this end, we deploy an untrained EdgeBOL in an environment where the wireless conditions quickly vary between 5 and 38 dB, as depicted by the first plot in Figure 3-60, and set $\delta_1 = 1$ and $\delta_2 = 8$. The top right plot depicts the size of the safe control set over time. As expected, the safe set quickly reduces within roughly 25 time periods and then adapts to the eventual contextual changes, with fluctuations matching the context changes. Remarkably, EdgeBOL convergences upon only 3 cycles across all contexts under evaluation. This is possible because the knowledge acquired by EdgeBOL for one context is actually transferred across similar contexts. That is, EdgeBOL is able to select judicious policies, shown in the remaining plots of the figure, even for unseen contexts. Specifically, for this choice of δ parameters, the GPU speed policy and the MCS policy highly vary upon context dynamics, whereas the image resolution and the airtime policy remain consistently high. It is worth mentioning that this policy dynamics is substantially different for diverse values of δ (not shown due to space constraints).

The above illustrates one of the major advantages of our approach, which makes appropriate decisions—even for contexts unseen before—by inferring correlations between the cost function and the context-control space. Conventional NN-based approaches are substantially less efficient in doing so, which renders EdgeBOL a particularly data-efficient solution.

To assess this, we implement a customized version of the deep deterministic policy gradient (DDPG) [48]. This benchmark is inspired by [49], which is the most related work to ours. Since the DDPG is designed to address the full-RL problem, we need to adapt it to address a contextual bandit problem, according to our

formulation. The DDPG algorithm uses an actor-critic NN architecture, but the critic, instead of approximating the Q function (full-RL problem), it learns a new cost function referred to as DDPG cost. The DDPG cost takes the value of (1) when all the constraints in (2) are satisfied, and the maximum cost value otherwise. Note that the DDPG does not handle constraints naturally and by using the DDPG cost function we allow the algorithm to do it. DDPG is particularly appealing for this type of problem because it operates with continued-valued control spaces. We mildly modified the architecture presented in [6] with a sigmoid function for the actor's output and optimized all the hyper-parameters (such as the decay) to minimize convergence time and performance.

We then test EdgeBOL and DDPG in a dynamic scenario where the constraint settings change over time: (i) $d^{\max} = 0.5\,\text{s}$, $\rho^{\min} = 0.4$ from $t = 0$ through $t = 1000$; (ii) $d^{\max} = 0.4\,\text{s}$, $\rho^{\min} = 0.6$ from $t = 1000$ through $t = 2000$; and (iii) $d^{\max} = 0.5\,\text{s}$, $\rho^{\min} = 0.5$ from $t = 2000$ on. Figure 3-61 depicts the evolution over time of the service delay and the mAP performance for both approaches. Not surprisingly, EdgeBOL rapidly converges to policies that respect the performance constraints, even when they suddenly change.

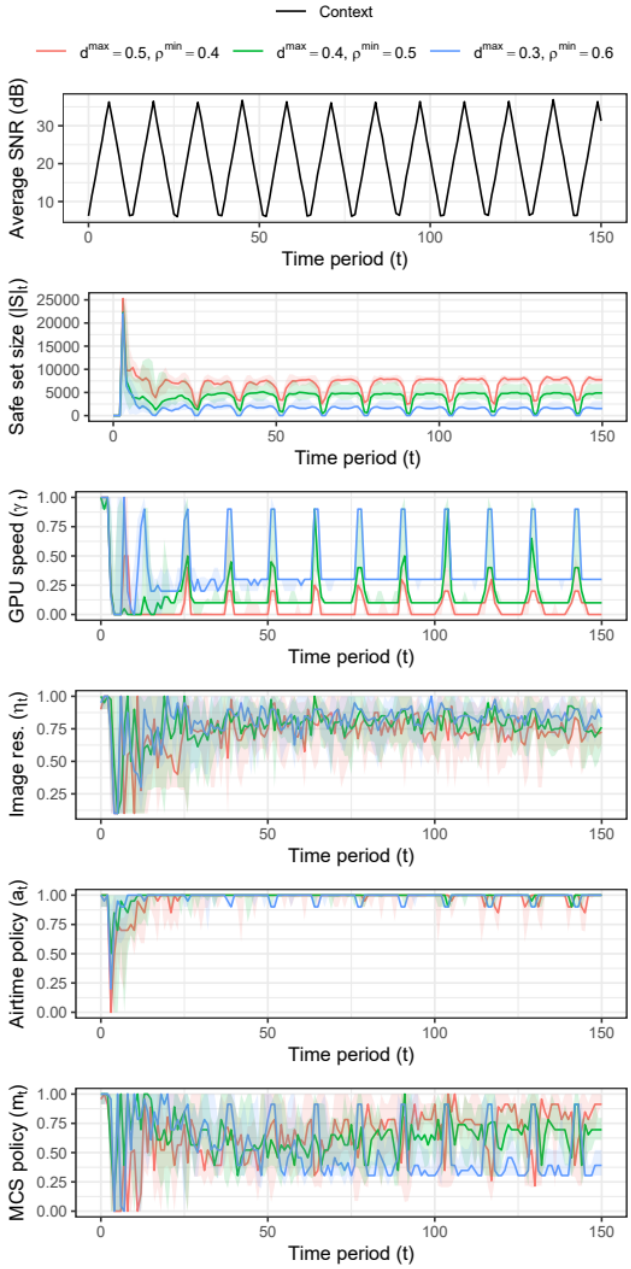


Figure 3-60: vRAN and Edge AI services - Evolution of policies for dynamic contexts (δ=8).

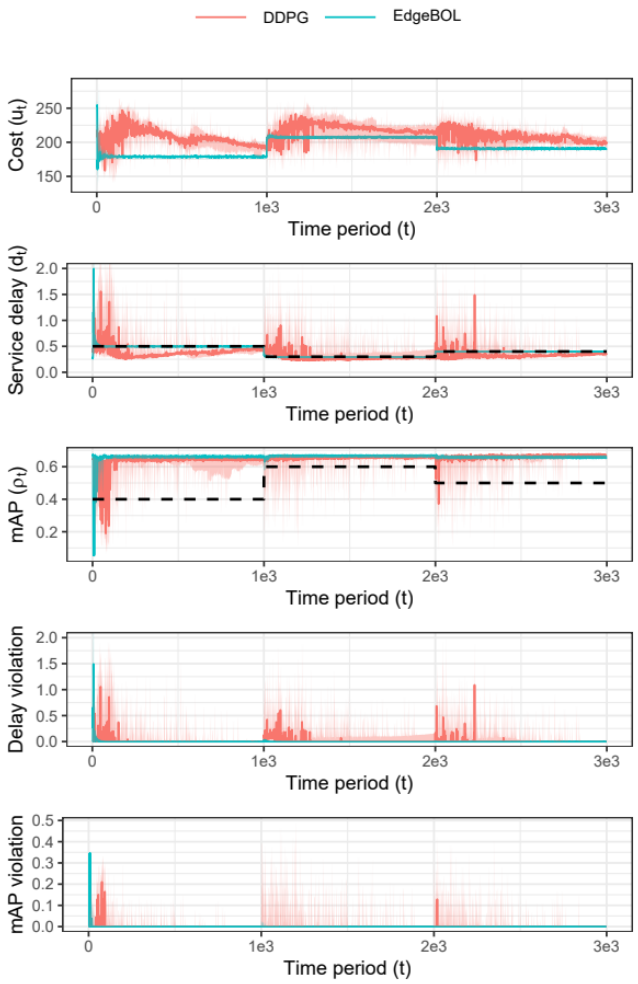


Figure 3-61: vRAN and Edge AI services - Evolution of delay and mAP upon changes on the constraint settings for EdgeBOL and a DDPG approach implemented with NNs ($\delta = 8$).

The non-parametric nature of our approach and the fact that we can compute safe control sets for any constrained setting based on prior data, allows EdgeBOL to drive the system to the new optimal points of operations almost instantaneously. In marked contrast, the NN-based benchmark takes a substantially higher number of time periods to find the new optimal—it is actually unable to converge prior to the constraint changes—and fails to adapt graciously upon constraint changes because NNs are parametric models that need to re-learn upon such changes.

3.5.5 Conclusions

The energy-aware implementation of AI services at the network edge is increasingly important for performance, economic and environmental reasons. The previous measurements showed non-trivial trade-offs between the delay and accuracy of such services, and revealed how these metrics are shaped by the base station and edge server control policies. In this final deliverable, using a Bayesian learning algorithm, the identification of a policy that minimises the aggregate energy costs minimised while adhering to predetermined performance criteria. This framework uses minimal assumptions and is proved effective in exploring the huge system configuration space. The proposed resource control mechanism is fully compliant with O-RAN and promising for enabling edge AI services, as it is verified experimentally using a prototype.

4 Summary and Conclusions

The BeGREEN D4.3 represents the culmination of efforts within Work Package 4, focusing on the final validation and evaluation of the BeGREEN Intelligence Plane and its AI/ML-driven solutions for enhancing energy efficiency in Beyond 5G (B5G) and 6G networks. This document builds upon the foundational work established in BeGREEN D4.1 and D4.2, refining the architecture, methodologies, and use cases to achieve significant energy savings without compromising network performance.

Key Achievements:

1. **Intelligence Plane Architecture**: The final architecture of the BeGREEN Intelligence Plane integrates advanced components such as the AI Engine, Non-RT RIC, and Near-RT RIC, along with the novel application of technologies like Reconfigurable Intelligent Surfaces (RIS), Integrated Sensing and Communication (ISAC), and cell-free distributed MIMO. This architecture enables cross-domain optimization and efficient management of RAN and Edge resources, ensuring seamless interoperability and scalability.
2. **Energy Efficiency Metrics**: The AI Engine's Energy Score and Energy Rating functions provide critical metrics for measuring and optimizing energy efficiency across network components. These tools allow for absolute and relative assessments of energy performance, supporting data-driven decision-making for energy-saving strategies.
3. **AI/ML Model Optimization**: Comprehensive benchmarks of AI/ML models during training and serving revealed important trade-offs between energy consumption, CPU usage, and model performance. Techniques such as feature reduction and CPU frequency management were shown to significantly reduce energy overhead while maintaining model accuracy. These techniques can be leveraged by the Model Selection function within the AI Engine to identify and provide the most energy-efficient model based on specified requirements.
4. **Validation of Solutions**: The document details the successful validation of key technologies, including:
    - **Intelligence Plane:** Complementing the work reported in D4.2, we characterized the performance of the AI Engine under different training and service workloads, providing insights on EE. We also validated some functionalities, such as the generation of datasets or the integration of RICs and the AI Engine, which will be further evaluated in the context of Work Package 5 demonstrations. The Near-RT RIC's conflict mitigation capabilities were validated, showing a 30% improvement in energy efficiency by resolving conflicts between energy-saving and QoS policies. Finally, the integration of RIS-enabled ISAC solutions with O-RAN architectures were also demonstrated, targeting enhanced indoor positioning and coverage extension
    - **vRAN:** Cache memory is a key resource for vRANs to reduce energy consumption. We proposed MemorAI which strategically allocates LLC resources to minimize energy consumption. MemorAI comprises a digital twin and a NN classifier, providing a very efficient and flexible solution. MemorAI achieves almost optimal performance and can attain significant energy savings when compared with other strategies.
    - **5G Carrier On/Off Switching**: Demonstrated the trade-offs between energy savings and Quality of Service (QoS) in a realistic 5G NSA deployment, with AI-driven strategies achieving nearly 79% energy savings under low QoS constraints, and over 20% under high QoS constraints.
    - **Relay-Enhanced RAN Control**: Showed that the deployment of fixed relays and their

dynamic activation/deactivation achieve power savings in the order of 15%-40%, depending on the bit rate and the power consumption model, with respect to the case of no relays. The use of relaying UEs to mitigate some coverage holes provides performance and power consumption results close to the case of fixed relays and reduces the deployment costs.

- **UPF Energy Efficiency**: Traffic-aware CPU resource management policies reduced energy consumption by 30% in realistic traffic scenarios, leveraging dynamic core and uncore frequency scaling. Traffic forecasting was shown to perform feasibly in this scenario, enabling the proactive determination of policies.
- **Joint Orchestration of vRAN and Edge AI Services**: Building on prior findings that highlighted trade-offs between delay and accuracy, the final deliverable introduces a Bayesian learning-based policy that minimizes energy consumption while meeting performance targets. This approach effectively navigates a large configuration space with minimal assumptions, aligns with O-RAN standards, and is experimentally validated using a prototype for edge AI services.

Impact and Future Directions:

The advancements presented in this deliverable highlight the commitment of BeGREEN to sustainable and intelligent network design. By addressing critical challenges in energy efficiency, the project paves the way for scalable, high-performance B5G and 6G networks that align with global environmental goals. Key takeaways include:

- The importance of balancing energy savings with QoS requirements to ensure user satisfaction.
- The potential of AI/ML-driven solutions to automate and optimize network operations dynamically.
- The need for continuous monitoring and adaptation to evolving network conditions and traffic patterns.

Future work will focus on further integrating these solutions into real-world deployments, exploring additional use cases, and refining the capabilities of the Intelligence Plane to support emerging technologies and standards. The insights and methodologies developed in BeGREEN WP4 provide a robust foundation for ongoing innovation in energy-efficient network management.

Through these efforts, BeGREEN drives sustainable advancements in next-generation communication systems, ensuring that performance and environmental responsibility go hand in hand.

5 Bibliography

[1] BeGREEN, D4.1, "State-of-the-Art Review and Initial Definition of BeGREEN O-RAN Intelligence Plane", December 2023. [Online] Available: https://www.sns-begreen.com/deliverables

[2] BeGREEN, D4.2, "Initial evaluation of BeGREEN O-RAN Intelligence Plane, and AI/ML algorithms for NFV user-plane and edge service control energy efficiency optimization", July 2024. [Online] Available: https://www.sns-begreen.com/deliverables

[3] O-RAN Alliance Working Group 1 (Use Cases and Overall Architecture), "O-RAN Architecture Description", O-RAN.WG1.OAD-R003-v12.00, June 2024.

[4] O-RAN Software Community, "Non-RT RIC Information Coordination Service", https://docs.o-ran-sc.org/projects/o-ran-sc-nonrtric-plt-informationcoordinatorservice/en/latest/overview.html Accessed Feb 2025

[5] O-RAN Alliance Working Group 6 (Cloudification and Orchestration Workgroup), "O-RAN Cloud Architecture and Deployment Scenarios for O-RAN Virtualized RAN 7.0", O-RAN.WG6.CADS-v07.00, June 2024

[6] O-RAN Alliance Working Group 6 (Cloudification and Orchestration Workgroup), "O-RAN O2 Interface General Aspects and Principles", O-RAN.WG6.O2-GA&P-R003-v07.00, June 2024.

[7] O-RAN Alliance Working Group 6 (Cloudification and Orchestration Workgroup), "O-RAN O2 Interface General Aspects and Principles", O-RAN.WG6.O2-GA&P-R003-v07.00, June 2024.

[8] 3GPP TS 38.401 V17.3.0, "NG-RAN; Architecture description (Release 17)", December, 2022.

[9] 3GPP TS 22.261 v19.5.0, "Service requirements for 5G system; Stage 1 (Release 19)", December, 2023.

[10] BeGREEN, D3.3., "Final evaluation and benchmarking of the implemented solutions", March 2024, To be published.

[11] O-RAN Alliance, *dApps for Real-Time RAN Control: Use Cases and Requirements*, O-RAN.nGRG-RR.2024.10, Apr. 2024. [Online]. Available: https://mediastorage.o-ran.org/ngrg-rr/nGRG-RR-2024-10-dApp%20use%20cases%20and%20requirements.pdf

[12] O-RAN Alliance, O-RAN Open F1/W1/E1/X2/Xn/D2 Interfaces Working Group, "D2 Interface: General Aspects and Principles", Technical Specification O-RAN.WG5.TS.D2GAP.0-R004-v01.00, Work in Progress, March 2025.

[13] NGMN Alliance, "NGMN 5G White paper", 5G Initiative Team, February 2015, Available online: https://ngmn.org/wp-content/uploads/NGMN_5G_White_Paper_V1_0.pdf

[14] J. Armstrong and E. Fallon, "Dimensionality Reduction for Optimization of Radio Base Station Transmission Based on Energy Efficiency," 2023 IEEE 64th International Scientific Conference on Information Technology and Management Science of Riga Technical University (ITMS), Riga, Latvia, 2023, pp. 1-6, doi: 10.1109/ITMS59786.2023.10317744

[15] Recursive Feature Selection, https://xgboosting.com/xgboost-feature-selection-with-rfe/

[16] K. Ahmed, M. Bollen, M. Alvarez, "A Review of Data Centers Energy Consumption And Reliability Modeling", IEEE Access, November 2021, 10.1109/ACCESS.2021.3125092

[17] Intel VTune, https://www.intel.com/content/www/us/en/developer/tools/oneapi/vtune-profiler.html

[18] Perf Tool, https://perf.wiki.kernel.org/index.php/Main_Page

[19] BeGREEN, “D2.3 Energy efficient RAN architecture and strategies”, March 2024, To be published.

[20] BeGREEN, D2.2 “Evolved Architecture and Power Enhancement Mechanisms”, [Online] Available: https://www.sns-begreen.com/deliverables

[21] G. Zhang, D. Zhang, H. Deng, Y. Wu, F. Zhan, and Y. Chen, “Practical passive indoor localization with intelligent reflecting surface,” IEEE Transactions on Mobile Computing, 2024

[22] H. Wymeersch, J. He, B. Denis, A. Clemente, and M. Juntti, “Radio localization and mapping with reconfigurable intelligent surfaces: Challenges, opportunities, and research directions,” IEEE Vehicular Technology Magazine, vol. 15, no. 4, pp. 52–61, 2020.

[23] O-RAN Alliance, “WG3, E2 Service Model (E2SM) KPM (O-RAN.WG3.E2SM-KPM-R003-v04.00),” Tech. Rep., 2023.

[24] X. Foukas, B. Radunovic, M. Balkwill, and Z. Lai, “Taking 5g ran analytics and control to a new level,” in Proceedings of the 29th Annual International Conference on Mobile Computing and Networking, 2023, pp. 1–16.

[25] RISE-6G, “RISE network architectures and deployment strategies analysis: final results),” H2020-ICT-52/RISE-6G/D2., Tech. Rep., 2023.

[26] dApps for Real-Time RAN Control: Use Cases and Requirements: https://mediastorage.o-ran.org/ngrg-rr/nGRG-RR-2024-10-dApp%20use%20cases%20and%20requirements.pdf

[27] O-RAN Alliance, “WG3, E2 Service Model (E2SM) KPM (O-RAN.WG3.E2SM-KPM-R003-v04.00),” Tech. Rep., 2023.

[28] RISE-6G, “RISE network architectures and deployment strategies analysis: final results),” H2020-ICT-52/RISE-6G/D2., Tech. Rep., 2023.

[29] srsRAN, “Open-source 4g and 5g software radio suites developed by software radio systems.” [Online; accessed 21-Nov-2024]. [Online]. Available: https://www.srsran.com/

[30] O-RAN Alliance, “O-RAN SC (I Release),” 2024. [On- line]. Available: https://lf-o-ran-sc.atlassian.net/wiki/spaces/REL/pages/12812314/I+Release

[31] 3GPP TR 38901, Study on channel model for frequencies from 0.5 to 100 GHz, (Release 18) v18.0.0 (2024-03).

[32] Siemens AG, “TDD UE-UE Interference Simulations”, R4-030189 document of the 3GPP TSG-RAN Working Group 4 meeting #26, February, 2003.

[33] O. Ruiz, J. Sánchez-González, J. Pérez-Romero, O. Sallent, I. Vilà, "Space and time user distribution measurements dataset in a university campus", Computer Networks, Volume 243, 2024, https://doi.org/10.1016/j.comnet.2024.110329.

[34] 3GPP TS 38.214 v17.0.0, "NR; Physical layer procedures for data (Release 17)", December, 2021.

[35] EARTH project, “D2.3 - Energy efficiency analysis of the reference systems, areas of improvements and target breakdown”, January, 2012.

[36] R. Fantini, D. Sabella, M. Caretti, “An E3F based assessment of energy efficiency of Relay Nodes in LTE-Advanced networks”, PIMRC, 2011.

[37] E. Björnson, M. Kountouris and M. Debbah, "Massive MIMO and small cells: Improving energy efficiency by optimal soft-cell coordination," ICT 2013, Casablanca, Morocco, 2013, pp. 1-5, doi: 10.1109/ICTEL.2013.6632074.

[38] G. Auer et al., "How much energy is needed to run a wireless network?", in IEEE Wireless Communications, vol. 18, no. 5, pp. 40-49, October 2011, doi: 10.1109/MWC.2011.6056691.

[39] FD.io Project. 2024. What is VPP? Accessed: 2024-12-06

[40] Julita Corbalan, Oriol Vidal, Lluis Alonso, and Jordi Aneas. 2021. Explicit uncore frequency scaling for energy optimisation policies with EAR in Intel architectures. In 2021 IEEE International Conference on Cluster Computing (CLUSTER). 572–581. https://doi.org/10.1109/Cluster48925.2021.00089

[41] Daniel Hackenberg, Robert Schöne, Thomas Ilsche, Daniel Molka, Joseph Schuchart, and Robin Geyer. 2015. An Energy Efficiency Feature Survey of the Intel Haswell Processor. In 2015 IEEE International Parallel and Distributed Processing Symposium Workshop. 896–904. https://doi.org/10.1109/IPDPSW.2015.70

[42] Intel Corporation. 2024. Running Average Power Limit (RAPL) Energy Reporting. https://www.intel.com/content/www/us/en/developer/articles/technical/software-security-guidance/advisory-guidance/running-average-power-limit-energy-reporting.html. Accessed: 2024-12-06.

[43] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. 2007. Power provisioning for a warehouse-sized computer. SIGARCH Comput. Archit. News 35, 2 (May 2007), 13–23. https://doi.org/10.1145/1273440.1250665

[44] Taylor, S. J., & Letham, B. (2018). Forecasting at Scale. The American Statistician, 72(1), 37–45. https://doi.org/10.1080/00031305.2017.1380080

[45] Felix Berkenkamp, Andreas Krause, and Angela P Schoellig. 2016. Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics. arXiv preprint arXiv:1602.04450 (2016).

[46] Marcello Fiducioso, Sebastian Curi, Benedikt Schumacher, Markus Gwerder, and Andreas Krause. 2019. Safe contextual Bayesian optimization for sustainable room temperature PID control tuning. arXiv preprint arXiv:1906.12086 (2019).

[47] Gomez-Miguelez, Ismael, et al. "srsLTE: An open-source platform for LTE evolution and experimentation." Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization. 2016.

[48] Lillicrap, Timothy P., et al. "Continuous control with deep reinforcement learning." arXiv preprint arXiv:1509.02971 (2015).

[49] Jose A Ayala-Romero, Andres Garcia-Saavedra, Marco Gramaglia, Xavier CostaPerez, Albert Banchs, and Juan J Alcaraz. 2020. vrAIn: Deep Learning based Orchestration for Computing and Radio Resources in vRANs. IEEE Transactions on Mobile Computing (2020).

Appendix 1: Characterization of the gNB and relay transmitted power

Let assume a specific macrocell gNB with cellular technology (e.g. 5G-NR) and *R* relays connected to this gNB. This work considers communication in the downlink direction, i.e., from the gNB/relays to the UEs, and assumes that the gNB and the relays operate at different frequencies. The UEs may connect directly to the gNB or through a relay (that may be either a fixed relay or a RUE). Let consider *K* UEs that connect directly to the gNB. Additionally, *R* relays are also connected to the gNB and each *r*-th relay (*r=1,…,R*) is serving J_r UEs.

In case of a *k*-th UE (*k=1,…,K*) directly connected to the gNB, the capacity per Resource Block (RB) in this link can be determined according to the Shannon bound as:

$$C_k = B_{RB,NB} \cdot \varepsilon_{NB} \cdot log_2\left(1 + \frac{P_{RB,NB} \cdot \beta_k}{P_N}\right) \quad \text{(A1-1)}$$

Where $P_{RB,NB}$ denotes the transmitted power per RB at the gNB, $P_N=N_0 \cdot B_{RB,NB}$ is the noise power measured over the RB bandwidth $B_{RB,NB}$, with N_0 as the noise power spectral density. The term ε_{NB} is an efficiency factor $0<\varepsilon_{NB}\leq 1$ that accounts for the overheads associated to cyclic prefix, reference signals, control plane signalling, etc. The term β_k in (8) corresponds to the gNB-UE link channel gain, considering the antenna gains and the propagation loss, defined as:

$$\beta_k = \frac{G_T \cdot G_T}{L_k} \quad \text{(A1-2)}$$

Where G_T and G_R denote the transmit and receive antenna gains, and L_k is the propagation loss between the gNB and the *k*-th UE, that includes the distance dependent loss and the slow fading (shadowing). In order to satisfy a specific UE bit rate requirement r_b, the number of required RBs at the gNB in the downlink direction can be calculated as:

$$M_{req,NB,k} = \left\lceil \frac{r_b}{C_k} \right\rceil \quad \text{(A1-3)}$$

Where the operator denotes the smallest integer higher than x.

When a *j*-th UE (*j=1,…,Jr*) is connected through a specific *r*-th relay (*r=1,…,R*), the number of RBs required at the gNB ($M_{req,NB,r}$) and at the *r*-th relay ($M_{req,r,j}$) can be determined following the same procedure as before, by means of equations (8) and (10):

$$M_{req,NB,r} = \left\lceil \frac{r_b}{B_{RB,NB} \cdot \varepsilon_{NB} \cdot log_2\left(1 + \frac{P_{RB,NB} \cdot \beta_r}{P_N}\right)} \right\rceil \quad \text{(A1-4)}$$

$$M_{req,r,j} = \left\lceil \frac{r_b}{B_{RB,R} \cdot \varepsilon_R \cdot log_2\left(1 + \frac{P_{RB,R} \cdot \beta_{j,r}}{P_{N,R}}\right)} \right\rceil \quad \text{(A1-5)}$$

Where $P_{RB,R}$ is the power per RB at the relay, $B_{RB,R}$ is the relay bandwidth per RB, and ε_R is the efficiency factor in the link between the relay and the UE. The noise power is determined as $P_{N,R}=N_0 \cdot B_{RB,R}$. Finally, the channel gains in the gNB-Relay link and the link between the *r*-th relay and the *j*-th UE is characterised by the terms β_r and $\beta_{j,r}$, respectively, according to the corresponding antenna gains and propagation losses.

Then, the total number of RBs required at the gNB is determined as:

$$M_{req,NB} = \sum_{k=1}^{K} M_{req,NB,k} + \sum_{r=1}^{R} J_r \cdot M_{req,NB,r} \quad \text{(A1-6)}$$

Note that the first term corresponds to the required number of RBs associated to the *K* UEs that are directly connected to the gNB, and the second term is the required number of RBs at the gNB associated to the gNB-Relay links. It is worth mentioning that, in order to satisfy the UE bit rate requirements, the total number of required RBs at the gNB must be below the maximum number of available RBs at the gNB (i.e. $M_{req,NB}<M_{NB}$). The total transmitted power associated to the gNB is determined as:

$$P_{T,NB} = P_{RB,NB} \cdot M_{req,NB} \quad \text{(A1-7)}$$

Similarly, for a specific *r*-th relay, the total transmitted power can be determined as:

$$P_{T,r} = P_{RB,R} \cdot \sum_{j=1}^{J_r} M_{req,r,j} \quad \text{(A1-8)}$$

As before, in order to satisfy the UEs bit rate requirements, the number of required RBs at the relay must be below the maximum number of RBs M_R available at the relay.